Application Control 2.x User Guide

Special notice

Before using this information and the product it supports, read the information in Notices *(on page clxii)*.

Edition notice

This edition applies to version 10.0 of BigFix and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Chapter 1. Application Control 2.x Overview **6**

Chapter 2. WDAC Rule Types and Use Cases **8**

Chapter 3. Application Control v2.x Best Practices **11**

Chapter 4. WDAC Policy Event ID Reference **16**

Chapter 5. WDAC Limitations **18**

Chapter 6. BitLocker Key Requirements **25**

Chapter 7. Policy Removal and Retirement **26**

Chapter 8. Boot-Stop Failure and Emergency Recovery **29**

Chapter 9. Behavior Considerations **32**

Chapter 10. Administering v2.x **35**

Root Server Installation 37

WDAC Management Node Installation 41

Certificate Provisioning (TPM) 44

Create and Manage WDAC Policies 49

Compile, Sign, and Publish 60

Set Up and Initialize WDAC Endpoints 64

Deploy and Refresh WDAC Policies 66

Uninstall Root Server BAC Configuration 70

Uninstall Management Node BAC Configuration 73

Uninstall Target Endpoint WDAC Components 75

Submit WDAC Policy for Review 79

Generate Custom WDAC Policy XML 82

Sync Published Policy XML to Management Node 85

Generate WDAC/AppLocker Blocked Events 87

Chapter 11. Deprecated Fixlets and Tasks **90**

AC v2.0.0 Use Cases 92

Migrate to v2.0.0 97

Create Self-Signed Cert on Endpoint 98

Deploy Default Microsoft Base Policy 101

Create Custom WDAC Base Policy 105

Deploy Custom Base Policy....109
Deploy Supplemental Policy on Endpoint....113
Refresh Self-Signed Cert and Update Policy Signers....116
Refresh Thumbprint of Deployed Policy Signers....119
Enforce Secure ACL on BAC Folder....121
Generate Blocked WDAC Event Logs....123
Remove Base or Supplemental WDAC Policy....126
Remove WDAC Components from Endpoint....129
View Endpoint Details in Web Reports....131
AC v2.0.0 Tasks Exit Code Reference....134
Chapter 12. FAQs....141
Chapter 13. Troubleshooting....156
Notices....clxii
Index....

Chapter 1. Application Control 2.x Overview

HCL BigFix Application Control 2.x uses Windows Defender Application Control (WDAC) to enforce application allowlisting on Windows endpoints. Use this topic to understand what Application Control does, what changed in the latest release, and how its three-tier architecture is deployed.

Application Control 2.x is a BigFix Lifecycle component that uses Windows Defender Application Control (WDAC) to enforce application allowlisting on managed endpoints. This overview provides a summary of the product, what is new in the current release, a reference to prior releases, and a description of the solution architecture.

Overview

Application Control 2.x enables IT administrators to define and enforce WDAC policies that control which applications and scripts are permitted to run on Windows endpoints. Managed through BigFix, it provides a scalable, policy-driven approach to application allowlisting — reducing attack surface without requiring per-device manual configuration. For a full introduction to the product, its goals, and supported configurations, see Overview.

What's New in This Release

Each release of Application Control 2.x introduces new Fixlets, Tasks, and policy management capabilities that extend WDAC coverage and simplify administration. The current release includes enhancements to policy compilation, certificate provisioning, and endpoint initialization workflows. For the full list of new features, resolved issues, and known limitations introduced in this release, see What's New in This Release.

Previous Releases

Application Control 2.x has evolved through a series of releases that progressively replaced the v1.x AppLocker-based approach with a WDAC-based architecture. Documentation for earlier releases — including deprecated Fixlets, migration guidance, and version-specific feature notes — is preserved for reference. For a chronological history of releases and links to archived release documentation, see Previous Releases.

User Personas

Application Control 2.x is designed for distinct roles within an IT organization, each with specific responsibilities in the policy lifecycle. These roles include BigFix administrators who deploy and maintain the BAC configuration, security engineers who author and sign WDAC policies on the Management Node, and endpoint administrators who initialize and manage target devices. Understanding which user persona applies to your role helps you navigate the relevant procedures and reference material. For a description of each user persona, their responsibilities, and the tasks they perform, see Personas.

System Architecture

The Application Control 2.x architecture is structured across three core tiers: a **BigFix Root Server** that stores and distributes BAC configuration content; a dedicated **WDAC Management Node** where policies are authored, compiled, and signed; and **targeted endpoints** where signed WDAC policies are deployed and enforced. Policy integrity is maintained through TPM-based certificate provisioning and digital signatures, guaranteeing that only

authorized policies are applied. For detailed component descriptions, data flows, and deployment details, see System Architecture.

Application Control 2.x Documentation

Overview

What's New in This Release

Previous Releases

System Architecture

Personas

Related Tasks

Administering Application Control v2.x *(on page 35)*

Application Control v2.x Best Practices *(on page 11)*

Chapter 2. BigFix Application Control: WDAC Rule Types, Selection Criteria, and Use Cases

BigFix Application Control supports six WDAC rule types — Publisher, Hash, File Path, File Attributes, Package/PFN, and COM — each suited to a different application identification method and control objective. This topic describes when to use each rule type, the benefits it provides, and illustrative example scenarios to guide rule selection in enterprise environments.

BigFix Application Control supports multiple WDAC rule types for identifying and controlling applications and application components. The appropriate rule type depends on **how the application or component should be identified and the level of control required**.

Table 1. WDAC Rule Types: Identification Method, Selection Criteria, and Benefits

Rule Type	When to Use	Benefit / Why Use It
Publisher	Use when trust is based on the **software publisher/signer**, especially for applications that receive signed updates.	Reduces the need to create new rules for applicable signed updates when the publisher/signing information remains trusted.
Hash	Use when you need to control a **specific executable, DLL, or file version**.	Simplifies control of applications and their files within a controlled location.
File Path	Use when an application runs from a **known and controlled local directory or network share**, and the location is part of the trust decision. **Note:** If a **user-writable location** must be allowed, **Option 18 – Disabled: Runtime FilePath Rule Protection** may be required, which reduces the protection provided by the default runtime check and has security implications.	Simplifies control of applications and their files within a controlled location.
File Attributes	Use when an application can be reliably identified using its **file metadata**, such as **OriginalFilename**, ProductName, or FileVersion. The required metadata, particularly **OriginalFile-**	Provides an alternative to hash- or path-based identification by using reliable file metadata, and can be used to target a specific application ver-

Table 1. WDAC Rule Types: Identification Method, Selection Criteria, and Benefits (continued)

Rule Type	When to Use	Benefit / Why Use It
	name, must be present in the file for the rule to match.	sion when the relevant version information is available.
Package / PFN	Use for applications distributed as **AppX/MSIX packages** that have package identity information, such as a PFN and also the executable is physically located inside the package. While creating rule, AC v2.x does not require binary to create rule.	Controls the packaged application using its package identity rather than individual executables.
COM	Use when a specific **COM component/class (CLSID)** needs to be separately controlled when another application or process attempts to activate it.	Provides separate control over COM activation, independent of the calling application's normal execution authorization.

Example Scenarios

The following software examples are **illustrative only**. Other applications can use the same rule type when the corresponding criteria apply.

- **Publisher:** Applications such as **Google Chrome, Mozilla Firefox, or Adobe applications** can use Publisher rules when trust is based on the publisher/signer and applicable signed updates should continue to be trusted.
- **Hash:** Applications such as **Notepad++, Postman, or AnyDesk** can use Hash rules when only a specific tested executable or version should be authorized.
- **File Path:** An **internal business application, WDAC Manager, or other application** can use a File Path rule when it runs from a known and controlled directory.
- **File Attributes:** Applications such as **Notepad++ or Chrome** can use File Attributes rules when file name, product name, version, or other supported metadata provides a suitable identification method.
- **Package / PFN:** Applications such as **Spotify, Microsoft Photos, or Calculator**, when installed as AppX/MSIX packages, can be identified by their package identity/PFN and the executable must be physically located inside the package.

  Traditional Win32 applications installed as standalone .exe/.msi applications, such as a conventional **Notepad ++ or Chrome installation**, generally do not have AppX/MSIX package identity and therefore require another applicable rule type.
- **COM:** An application such as **Notepad++** may have a COM-based component. If another application or process attempts to activate that component using its **CLSID**, a COM rule can be used to explicitly allow or block that activation.

Note: The applications listed above are illustrative examples only. An application may be controlled using different rule types depending on its signing information, installation method, file characteristics, package identity, COM dependencies, and the organization's security requirements. The selected rule type should therefore be based on the **application control objective and the characteristics of the application or component**, rather than on the application name alone.

Product Context

BigFix Application Control: Product Overview

WDAC Limitations *(on page 18)*

Behavior Considerations *(on page 32)*

Policy Authoring and Management

Managing WDAC Policies on the Management Node *(on page 49)*

Creating a New Base WDAC Policy *(on page 105)*

Deploying a Supplemental Policy *(on page 113)*

Compiling, Signing, and Publishing WDAC Policies *(on page 60)*

Troubleshooting and FAQs

Frequently Asked Questions *(on page 141)*

Troubleshooting Application Control *(on page 156)*

Chapter 3. BigFix Application Control v2.x: Best Practices

Guidance for deploying and managing WDAC policies through BigFix® Application Control 2.x — covering core validation principles, policy naming and overwrite behavior, change tracking recommendations, dual-node emergency signing and certificate rotation, co-located management node constraints, and the recommended rule validation workflow for onboarding new applications.

Best Practices for WDAC Policy Deployment

To ensure robust endpoint security without disrupting business operations or user productivity, adhere to these core deployment principles, architectural guidelines, validation workflows, and operational procedures across all managed environments:

Core Principles:

- **Validate Before Enforcement:** Always deploy new policies, policy changes, or supplemental rules in **Audit Mode** within a designated **User Acceptance Testing (UAT)** environment before enforcing them.
- **Data-Driven Rule Creation:** Use Code Integrity event logs (`3076`, `3034`) collected during UAT to identify required application components and select the most maintainable rule criteria — prioritizing **Publisher/Signer** rules over **Hash** or **Path** rules where feasible.
- **Zero-Impact Promotion:** Promote a policy to **Enforcement Mode** only after iterative UAT testing confirms all core application features, background tasks, dependent modules, and update workflows execute without generating audit events.
- **Controlled Phased Rollout:** Following successful UAT sign-off, deploy validated policies to production endpoints in progressive rollout rings adhering to your organization's standard change management process and Change Advisory Board (CAB) approval guidelines.
- **Emergency Break-Glass & Dual-Node Provisioning:** Pre-establish a dual-node emergency signing lifecycle (Node I Primary, Node II Emergency) to guarantee policy administration continuity, seamless failover, and certificate rotation without breaking kernel enforcement on managed endpoints.

Important Advisory: Policy Naming and Overwrite Behavior

- **Base and Supplemental Policy Overwrites:** Creating a new Base or Supplemental policy using an existing policy name will permanently overwrite the original XML policy file.
- **Deployment Site Impact:** Publishing a policy with a duplicate name overwrites the existing files in the Deployed site. Deploying this updated policy to endpoints will immediately apply the new policy rules, potentially altering or reverting active enforcement on target endpoints.
- **Web Reports & History Tracking:** Overwriting a Supplemental policy bypasses the Fixlet purge mechanism. As a result, Web Reports will not track historical changes or show active/inactive states for overwritten policies.

Recommendations to Preserve Change Tracking

- **Assign Unique Names:** Always assign unique names when creating new Base or Supplemental policies unless you explicitly intend to replace the existing policy and overwrite its history.
- **Preserving Web Reports History:** To block a previously allowed rule while keeping full historical records in Web Reports, use the **Purge Supplemental Policy Rule** Fixlet instead of overwriting the policy. This preserves the old policy file and displays the historical record in Web Reports.

Emergency Signing and Certificate Rotation Synchronization Process

This lifecycle governs policy administration when the Primary Signing Certificate expires, rotates, or fails due to infrastructure issues. Emergency signing is strictly audited and restricted to authorized recovery procedures.

Dual-Node Emergency Signing and Rotation Flow

- **Mutual Trust Baseline:** Public certificates (`.cer`) for **Node I (Primary)** and **Node II (Emergency)** are defined as authorized signers in the Base Policy XML, allowing endpoints to accept signed updates from either node without breaking kernel enforcement.
- **Emergency Authorization:** An administrator detects primary signing failure or certificate expiration and formally authorizes emergency protocol execution per security governance.
- **Emergency Signing (Node II):** Node II is temporarily subscribed to the Central Custom Site Repository, signs the active policy XML using its local hardware-protected Emergency Signing Certificate, and unsubscribes to minimize attack surface boundaries.
- **Deployment Handoff:** The emergency-signed `.cip` binary routes through the Upload Manager and Root Server to the Custom Site for continuous fleet deployment.
- **Primary Certificate Re-provisioning:** A new Primary Signing Certificate is generated, bound to TPM 2.0 on Node I, and its public key is shared with Node II for trust synchronization.
- **State Sync & Failback (Node I):** Node I temporarily subscribes to the Central Repository to pull the latest policy state, updates its definitions to trust the new primary certificate while retaining Node II's emergency certificate, re-signs the policy via TPM, and unsubscribes.
- **Fleet Re-alignment:** BigFix® Agents deploy the newly signed Base policy to managed endpoints, restoring primary governance without interrupting kernel enforcement. Compliance is verified via BigFix® Web Reports.

Co-located Management Node Setup (Not Recommended)

Architectural Recommendation: Deploying a dedicated, isolated WDAC Management Node is strongly recommended to keep policy-signing capabilities separated from general endpoints. Co-locating management components on an enforced endpoint introduces significant operational risk and is **not advisable**.

If operational constraints strictly require applying policy enforcement on the same host running the WDAC Management components, follow this mandatory workflow:

- **Keep Base Policies Clean:** Never add allow rules for `signtool.exe` or `wdacmanager.exe` to the global enterprise Base Policy, as this exposes code-signing capabilities fleet-wide.
- **Pre-compile the Supplemental Policy:** Create, compile, and sign the WDAC Management Supplemental Policy (authorizing `wdacmanager.exe` and `signtool.exe`) before applying a restrictive Base Policy to the host.
- **Stage to Both Active Paths:** Ensure both `.cip` policy binaries are deployed to both required storage locations:
  - **OS Drive:** `%OSDrive%\Windows\System32\CodeIntegrity\CiPolicies\Active\{PolicyID}.cip`
  - **EFI Partition:** `<EFI Partition System>\Microsoft\Boot\CiPolicies\Active\{PolicyID}.cip`

Recommended Rule Validation Approach

Note: Audit Mode is intended for validation and does not provide the same blocking behavior as Enforcement Mode. Event data should be evaluated together with the application's signing information, file characteristics, installation path, package identity, and dependencies when determining the appropriate rule type. An audited component should not automatically result in a broader allow rule; the rule should be selected based on the application's characteristics and the intended application control objective.

Follow this validation process when deploying an Application Control policy, onboarding a new application into WDAC, or determining the appropriate rule criteria for application dependencies.

Validation Workflow:

- **Deploy in Audit Mode:** Apply the WDAC policy in **Audit Mode** on designated UAT test endpoints.
- **Execute Workflows:** Run the application and perform all standard business workflows in UAT, including background services, secondary modules, and update routines.
- **Monitor Code Integrity Events:** Track generated WDAC/Code Integrity events to identify audited execution attempts.
- **Aggregate & Review Events:** Review and analyze collected event logs using BigFix® Web Reports or your enterprise reporting framework.
- **Adjust & Refine Rules:** Evaluate observed file attributes (Publisher, Hash, Path, or File Name) and generate required supplemental rules or adjust rule types accordingly (detailed in Annexure 1 *(on page 13)*).
- **Iterative Verification:** Repeat steps 1—5 until all application functionality executes in UAT without generating audit events.
- **Verify in Enforcement Mode:** Apply the updated policy in **Enforcement Mode** on the UAT machine to confirm full operational capability under active enforcement.
- **Production Rollout:** Upon successful UAT sign-off, deploy the finalized policy to production endpoints according to your organization's standard change management process.

Annexure 1: Creating a Supplemental Policy via App Control Policy Wizard

When WDAC policy events occur, details are captured in the target endpoint's Windows Event Viewer under **Applications and Services Logs > Microsoft > Windows > CodeIntegrity > Operational**.

Key Code Integrity Event IDs:

- **Audit Mode Events (used during validation):**
  - `3076` — User-mode Audit event (indicates an executable or DLL would have been blocked in Enforcement Mode).
  - `3034` — Kernel-mode / Driver Audit event (indicates a driver would have been blocked in Enforcement Mode).
- **Enforcement Mode Block Events (active policy enforcement):**
  - `3077` — User-mode Block event (indicates an executed file was blocked by policy).
  - `3033` — Kernel-mode / Driver Block event (indicates a driver was blocked by policy).

Export Event Logs from UAT Endpoint

1. Open **Event Viewer** (`eventvwr.msc`) on the UAT test machine.
2. Navigate to **Applications and Services Logs > Microsoft > Windows > CodeIntegrity > Operational**.
3. Filter or select the relevant logs (Event IDs `3076` and `3034` for Audit Mode).
4. Right-click **Operational** and select **Save All Events As...** to export the `.evtx` log file.

Import Logs into App Control Policy Wizard

1. Launch the **App Control Policy Wizard** (see ***Microsoft App Control Policy Wizard*** from the Related Links *(on page 14)* section below).
2. Select **Policy Editor** on the home screen.
3. Select **Convert Policy Logs to a Policy** (or **Supplemental Policy** from **Event Log**).
4. Browse and select the exported `.evtx` log file.

Configure Supplemental Rules and Export

1. Review the list of audited/blocked files parsed by the wizard.
2. Assign the appropriate **Rule Type** (for example, **Publisher**, **File Publisher**, **Hash**, or **Path**) for each file entry based on organizational security guidelines.
3. Proceed through the wizard to build and output the supplemental policy XML.
4. Deploy the supplemental XML policy using the Generate Custom WDAC Policy XML task.

Rule Validation and Event Reference

WDAC Code Integrity and App Control Event ID Reference *(on page 16)*

Generate WDAC Code Integrity and AppLocker Blocked Events *(on page 87)*

Microsoft App Control Policy Wizard

Policy Management Tasks

Create and Manage WDAC Policies on the WDAC Management Node *(on page 49)*

Generate a Custom WDAC XML Policy on the WDAC Management Node *(on page 82)*

Compile, Sign, and Publish WDAC Policies from the WDAC Management Node *(on page 60)*

Submit a WDAC XML Policy for Review *(on page 79)*

Emergency Signing and Recovery

Provision the TPM-Backed Code-Signing Certificate on the WDAC Management Node *(on page 44)*

WDAC Policy Boot-Stop Failure and Emergency Recovery *(on page 29)*

Conceptual Reference

BigFix Application Control v2.x: Policy Behavior and Deployment Considerations *(on page 32)*

WDAC Rule Types and Use Cases *(on page 8)*

Standard WDAC Policy Removal and Retirement Procedure *(on page 26)*

Chapter 4. WDAC Code Integrity and App Control Event ID Reference

Reference table of WDAC Code Integrity and App Control event IDs covering audit-mode and enforcement-mode events for binaries, scripts, packaged applications, and signature validation failures — used to identify blocked or audited application files on managed Windows endpoints.

The following table lists the WDAC Code Integrity and App Control event IDs that appear in the Windows Event Log when a policy is active in Audit Mode or Enforcement Mode. For each event type, the table identifies the audit-mode event ID (the file *would have been* blocked), the enforcement-mode event ID (the file *was* blocked), the target file scope, and a description of the event. Use these event IDs to identify audited or blocked application files when reviewing event data collected through BigFix® Web Reports or the Application Control reporting mechanism.



Note: Event 8040 has no audit-mode equivalent. Packaged application blocks (`MSIX`, `AppX`) are only surfaced as enforcement-mode events and are not visible in Audit Mode event logs.

Table 2. WDAC Code Integrity and App Control Event IDs — Audit Mode and Enforcement Mode

Audit Event	Enforced Event	Target Scope	Description
3076	3077	Binaries (`.exe`, `.dll`, `.sys`)	Main App Control file evaluation event. 3076 indicates that the file would have been blocked if the policy were enforced; 3077 indicates that the file was blocked by the enforced policy.
8028	8029	Scripts & Installers (`.ps1`, `.msi`)	App Control evaluation events for scripts and MSI installers. 8028 indicates that the file would have been blocked if the policy were enforced; 8029 indicates that the file was blocked by the enforced policy.
—	8040	Packaged Apps (`MSIX`, `AppX`)	Indicates that a packaged application was blocked by the App Control policy.
3034	3033	Code Integrity / Signature Validation	Indicates that the file under validation did not meet the requirements to pass Code Integrity/App Control. 3033 indicates an enforced failure, while 3034 is the audit-mode equivalent. The event identifies the validation failure, but the specific cause should be investigated using the associated Code Integrity and signature information, particularly Event 3089, together with the applicable policy and signing details.
3099	3099	App Control Policy Information	Provides information about the App Control policy processed by the system, including the Policy ID, which can be used to identify the policy processed.

Event Collection

Generate WDAC Code Integrity and AppLocker Blocked Events *(on page 87)*

View Endpoint Details from BigFix Web Reports *(on page 131)*

Policy Analysis and Rule Adjustment

Best Practices for WDAC Policy Rule Validation *(on page 11)*

Create and Manage WDAC Policies on the WDAC Management Node *(on page 49)*

Submit a WDAC XML Policy for Review *(on page 79)*

Conceptual Reference

BigFix Application Control v2.x: Policy Behavior and Deployment Considerations *(on page 32)*

WDAC Rule Types and Use Cases *(on page 8)*

Chapter 5. BigFix Application Control: WDAC and App Control for Business Known Limitations

BigFix Application Control deployments are subject to known Windows Defender Application Control (WDAC) / App Control for Business limitations documented by Microsoft. This reference covers platform and version requirements, policy structure constraints, signed-policy signing rules, Memory Integrity (HVCI) interactions, and Smart App Control conflicts — each with an impact assessment and a Microsoft documentation reference.

The following limitations are documented by Microsoft for WDAC / App Control for Business. Each limitation includes its impact on deployment and, where applicable, the minimum Windows version or configuration required. These limitations must be considered when designing, deploying, and managing Application Control policies in BigFix-managed environments.

Platform and Version Limitations

1. **Multiple Policy Support**

   Limitation: Multiple App Control policies can be deployed side-by-side only on the following versions:

   - Windows 10, version **1903 and later**
   - Windows 11
   - Windows Server **2022 and later**

   Impact: Supplemental policy and other multiple-policy scenarios require a supported Windows version. Deployments targeting older versions cannot use supplemental policies.

   Microsoft reference: Use multiple App Control for Business policies — Microsoft Learn

2. **Maximum Active Policy Count**

   Limitation:

   - Systems without the applicable April 9, 2024 or later security update are limited to **32 active policies**.
   - The 32-policy limit was removed by the applicable security update released **April 9, 2024 or later**.
   - **Windows 11 version 21H2 remains limited to 32 active policies** regardless of security update status.

   Microsoft reference: Use multiple App Control for Business policies — Microsoft Learn

3. **App Control Feature Availability Varies by Windows Version**

   Limitation: Not every App Control capability is available on every supported Windows version. The following table summarizes capability availability:

App Control Capability	Supported Starting Point
Multiple policies	Windows 10 1903+, Windows 11, Server 2022+
Path-based rules	Windows 10 1903+, Windows 11, Server 2022+
Managed Installer	Windows 10, Windows 11, Server 2019+
COM object allowlisting	Windows 10, Windows 11, Server 2019+
Application ID tagging	Windows 10 20H1+, Windows 11
Custom Kernel Signers	Windows 11 24H2+

Microsoft reference: App Control for Business feature availability — Microsoft Learn

4. Device-Wide Policy Scope

Limitation: App Control policies operate at the **device level**. App Control does not provide per-user or per-user-group policy scoping, unlike AppLocker.

Impact: Organizations that require per-user application control policies must use AppLocker or a hybrid approach, as WDAC cannot restrict or allow applications differently for different users on the same device.

Microsoft reference: App Control for Business feature availability — Microsoft Learn

5. Supported Enforceable File Types

Limitation: App Control enforcement applies only to specific file types. File types not in the supported list are not subject to App Control enforcement. Microsoft documents the following supported file types:

- `.sys`
- `.exe`
- `.com`
- `.dll`
- `.rll`
- `.ocx`
- `.msi`
- `.mst`
- `.msp`
- `.ps1`
- `.vbs`
- `.js`
- `.appx`

Microsoft reference: App Control for Business feature availability — Microsoft Learn

Policy Structure and Relationship Limitations

1. **Supplemental Policy Relationship**

   Limitation: A supplemental policy can expand **only one base policy**. Multiple supplemental policies can expand the same base policy, but nesting of supplemental policies is not supported.

   The supported relationship is:

   ```
   Base Policy
   ├── Supplemental Policy 1
   ├── Supplemental Policy 2
   └── Supplemental Policy 3
   ```

   A supplemental policy cannot itself act as the base policy for another supplemental policy.

   Microsoft reference: Creating a new Supplemental Policy — Microsoft Learn

2. **Base Policy Must Allow Supplemental Policies**

   Limitation: The base policy must be explicitly configured to allow supplemental policies before any supplemental policy can be applied to it.

   Impact: Deploying a supplemental policy against a base policy that has not been configured to permit supplemental policies will fail. The base policy must be authored or updated accordingly before supplemental policies can be used.

   Microsoft reference: Create a base App Control policy — Microsoft Learn

3. **Hash-Based Rules**

   Limitation: A hash rule identifies a specific file by its cryptographic hash. If the file is updated or modified, its hash changes and the existing hash rule no longer matches the modified file.

   Impact: Hash rules require ongoing maintenance whenever approved applications are updated. Microsoft also documents performance considerations for policies that contain very large numbers of hash rules.

   Microsoft reference: Understand App Control policy rules and file rules — Microsoft Learn

4. **Path-Based Rule Restrictions**

   Limitation: Path-based rules are available only on:

   - Windows 10 version **1903 and later**
   - Windows 11
   - Windows Server **2022 and later**

   Microsoft documents that **path exclusions are not supported**, and runtime user-writeability checks are enforced by default. A path that is user-writable at runtime will not be trusted by a path-based rule.

Microsoft reference: App Control for Business feature availability — Microsoft Learn

5. **Script Enforcement Version Restrictions**

   Limitation: The `Disabled:Script Enforcement` policy option is **not supported** on Windows Server 2016 or Windows 10 version 1607 LTSB.

   Impact: Using this option on those platforms can result in unexpected script-enforcement behavior. Deployments targeting these operating systems must not use this policy option.

   Microsoft reference: App Control for Business feature availability — Microsoft Learn

Signed Policy Limitations

1. **Signed Base Policy Replacement**

   Limitation: A signed base policy cannot simply be deleted. To replace a signed base policy, the replacement policy must satisfy all of the following conditions:

   - Use the **same Policy ID** as the policy being replaced.
   - Have a version **equal to or greater than** the existing policy.
   - Include the required policy configuration for removal/replacement.

   Microsoft reference: Remove App Control for Business policies — Microsoft Learn

2. **Signed Policy Version Requirement**

   Limitation: When replacing a signed policy, the replacement policy must have a version **equal to or greater than** the existing active policy. Deploying a lower-versioned signed policy is not permitted.

   Microsoft reference: Remove App Control for Business policies — Microsoft Learn

3. **Signed Policy Signing and Trust Requirements**

   Limitation: Signed App Control policies require appropriate signing and trust configuration. The signing certificate must be trusted by the policy itself, and signed supplemental policies must have the correct signing relationship with their base policy.

   Impact: Signed-policy deployment requires proper certificate and signing configuration to be established in advance. A policy file cannot be treated as signed merely by converting or deploying it; the signing relationship must be explicitly configured.

   Microsoft reference: Use signed policies to protect App Control for Business against tampering — Microsoft Learn

4. **Signed Base Policy Requires Existing Supplemental Policies to be Signed**

   Limitation: When signing a base policy that already has supplemental policies, all existing supplemental policies must also be converted to signed policies. The base policy must authorize the certificates used to sign the supplemental policies through the `<SupplementalPolicySigner>` rule.

Impact: Existing unsigned supplemental policies cannot remain unchanged when the base policy is signed. A coordinated signing of both the base and all associated supplemental policies is required when transitioning to a signed base policy.

Microsoft reference: Use signed policies to protect App Control for Business against tampering — Microsoft Learn

5. **Signed Policies Require Secure Boot for Anti-Tampering Protection**

   Limitation: Microsoft states that UEFI Secure Boot must be enabled on the endpoint for the anti-tampering protection provided by signed App Control policies to take effect. The protection takes effect after the first reboot following policy deployment.

   Impact: A signed-policy implementation must not assume that signing alone provides the documented anti-tampering protection. Endpoints must also have Secure Boot enabled; otherwise the anti-tampering guarantee does not apply.

   Microsoft reference: Use signed policies to protect App Control for Business against tampering — Microsoft Learn

6. **Signing Algorithm and Key Restrictions**

   Limitation: Microsoft specifies the following requirements for signing App Control policies:

   - **RSA only** — ECDSA is not supported.
   - RSA key sizes: **2048, 3072, or 4096 bits**.
   - SHA-256, SHA-384, or SHA-512 can be used on Windows 11 and on Windows 10 / Server 2019+ with the November 2022 cumulative update.
   - All other supported devices support SHA-256 only.
   - Policies must be signed according to **PKCS #7**.

   Impact: The signing implementation — including TPM-backed certificates — must generate certificates and signatures that comply with these algorithm and key-size requirements. Non-compliant certificates will result in policy rejection.

   Microsoft reference: Use signed policies to protect App Control for Business against tampering — Microsoft Learn

Memory Integrity (HVCI) and Hardware Limitations

1. **Memory Integrity (HVCI) and New Signed Base Policies**

   Memory Integrity is a Windows security feature based on Virtualization-Based Security (VBS), also known as Hypervisor-Protected Code Integrity (HVCI). It moves kernel-mode Code Integrity validation into an isolated environment protected by the Windows hypervisor.

   Availability: Memory Integrity is available on Windows 10, Windows 11, and Windows Server 2016 and later. Whether it is enabled by default depends on the Windows version, hardware, installation type, and

device configuration. Microsoft states that Memory Integrity is enabled by default on clean installations of Windows 11 on compatible hardware and on Secured-core PCs.

WDAC-specific limitation: Microsoft documents a known issue with deploying **new signed App Control base policies** on systems where Memory Integrity is enabled.

For **Windows 11 updates earlier than 24H2**, Microsoft recommends:

1. Deploy the new signed base policy through a **script** rather than Group Policy or MDM in the affected scenario.
2. Do **not** use CiTool, RefreshPolicy.exe, or WMI to initiate activation for this specific scenario.
3. Activate the policy by performing a **system restart**.

This issue **does not affect**:

- Updates to an already-active signed base policy.
- Unsigned policies.
- Supplemental policies, whether signed or unsigned.
- Systems where Memory Integrity is not enabled.

Microsoft references: Deploy App Control policies using script — Microsoft Learn; Deploy App Control policies via Group Policy — Microsoft Learn; Memory Integrity and VBS enablement — Microsoft Learn

2. Custom Kernel Signers

Limitation: Custom Kernel Signers are available only on:

- **Windows 11 version 24H2 and later**
- With the applicable **April 2026 non-security update** applied

The feature requires Secure Boot and has additional signing and platform requirements. It is **not supported on ARM64 systems with System Guard Secure Launch**.

Microsoft reference: Custom Kernel Signers for App Control for Business — Microsoft Learn

3. Secure Boot Requirements for Custom Kernel Signers

Limitation: Custom Kernel Signers require specific platform security configuration, including **Secure Boot**, and have additional firmware and signing requirements. This feature is supported only on applicable Windows 11 24H2+ systems.

Impact: Custom Kernel Signer scenarios cannot be treated as a standard WDAC policy deployment. Each target endpoint must meet the required Secure Boot, firmware, Windows-version, and signing prerequisites before this feature can be used.

Microsoft reference: Custom Kernel Signers for App Control for Business — Microsoft Learn

Smart App Control (SAC) Conflict

1. **Smart App Control — Existing App Control Policy Conflict**

   Limitation: On supported Windows 11, version 22572 or later, Smart App Control (SAC) can be enabled only after a **clean Windows installation**. SAC may start in Evaluation mode and, in most cases, automatically activates after Windows determines that the device is suitable.

   When SAC is active, it deploys its own App Control policy. The active SAC `.cip` policy file is located at:

   ```
   C:\Windows\System32\CodeIntegrity\CiPolicies\Active\
   ```

   Impact: SAC must be explicitly checked and handled before provisioning a custom WDAC/App Control base policy. The implementation must not assume that the endpoint is free of an existing App Control policy, nor that deploying a custom base policy will automatically replace or remove the SAC policy.

   Microsoft references: Smart App Control overview — Microsoft Learn; Application Control for Windows — Microsoft Learn; Test App Signatures with Smart App Control — Microsoft Learn

---

Product Context

BigFix Application Control: Product Overview

System Architecture

WDAC Rule Types and Use Cases *(on page 8)*

Behavior Considerations *(on page 32)*

Certificate and Signing Setup

Provisioning the TPM-Backed Code-Signing Certificate *(on page 44)*

Compiling, Signing, and Publishing WDAC Policies *(on page 60)*

Policy Deployment Tasks

Setting Up and Initializing WDAC on Target Endpoints *(on page 64)*

Deploying and Refreshing WDAC Policies on Endpoints *(on page 66)*

Deploying a Supplemental Policy *(on page 113)*

Removing Base and Supplemental Policies *(on page 126)*

Troubleshooting and FAQs

Frequently Asked Questions *(on page 141)*

Troubleshooting Application Control *(on page 156)*

Chapter 6. BitLocker Key Requirement & Consideration

On BitLocker-encrypted endpoints, disabling UEFI Secure Boot alters the TPM platform integrity measurement and may trigger a BitLocker recovery key prompt. Have the 48-digit BitLocker Recovery Key available before beginning any WDAC recovery procedure, and be prepared to temporarily suspend BitLocker protection during boot configuration changes.

BitLocker Recovery Key Requirement

Disabling UEFI Secure Boot changes the platform integrity state measured by the TPM and may cause BitLocker to require the recovery key. On BitLocker-protected devices, the applicable **48-digit BitLocker Recovery Key** should therefore be available before performing the recovery procedure.

BitLocker Consideration During Recovery

Depending on the recovery procedure and platform configuration, **BitLocker protection may need to be temporarily suspended during recovery** to allow the required boot configuration changes. Once recovery is complete and UEFI Secure Boot has been re-enabled, **resume BitLocker protection** to restore the normal security posture.

Recovery Prerequisites and Context

Provision the TPM-Enforced WDAC Policy Signing Certificate on the WDAC Management Node *(on page 44)*

Configure and Install the WDAC Management Node *(on page 41)*

Reference

BigFix Application Control: Troubleshooting WDAC Enforcement Issues *(on page 156)*

BigFix Application Control v2.x: Policy Behavior and Deployment Considerations *(on page 32)*

Chapter 7. Standard WDAC Policy Removal and Retirement Procedure

Retire active WDAC policies using a controlled two-stage sequence that neutralizes the policy's enforcement state before removing its artifacts from disk. This in-band procedure maintains endpoint execution continuity throughout the planned policy retirement lifecycle.

Overview

Standard policy retirement is an in-band administrative workflow performed through a BigFix® Fixlet or Task while Windows and the BigFix Agent are fully operational. The procedure follows the Microsoft App Control Removal Specifications and is designed around a deliberate two-stage sequence:

- **Stage 1 – Policy State Neutralization:** Deactivate the policy's enforcement state before touching any artifacts on disk.
- **Stage 2 – Policy Artifact Removal:** Delete the compiled policy binary (`.cip`) from all active policy storage locations and restart the endpoint to clear the policy state from system memory.

The appropriate Stage 1 workflow depends on the policy type being retired — signed Base policies and Supplemental policies follow different paths.

Stage 1, Option A — Signed Base Policy Replacement (Recommended)

For a signed Base Policy, deploy a signed replacement policy before removing any policy artifacts from disk. The replacement policy must meet all of the following requirements.

Policy ID and Version Alignment

The replacement policy must use the exact same `PolicyID` as the active policy and specify a version equal to or greater than the active policy (`VersionEx`).

Policy Configuration

Configure the replacement policy as an effectively inactive policy: use Allow * rules for files, allow all COM objects, enable Option 3 (Enabled:Audit Mode) and Option 11 (Disabled:Script Enforcement), include Option 6 (Enabled:Unsigned System Integrity Policy), and remove Option 0 (Enabled:UMCI) where applicable. This configuration is consistent with Microsoft's policy removal guidance.

Signer Continuity and Transitioning

The replacement policy must contain `<UpdatePolicySigners>` and be signed by a certificate already trusted by the active policy.

If transitioning to a new signer, first update the active policy — using an existing trusted signer — to authorize the new certificate in its `<UpdatePolicySigners>` block. The replacement policy can then be signed directly with the newly authorized certificate.

Mandatory Endpoint Restart

After deploying the replacement policy, restart the endpoint so that UEFI protection associated with the original signed policy is deactivated before proceeding to Stage 2.

⚠ **Warning:** Do not remove the original policy artifacts before restarting the endpoint after deploying the replacement policy. Failing to restart first — while UEFI protection for the original signed policy is still active — will result in a boot-start failure (BSOD).

Stage 1, Option B — Supplemental Policy Removal

For a Supplemental Policy — signed or unsigned — the signed Base Policy replacement workflow is not required. Microsoft explicitly states that signed Supplemental Policies can be removed in the same manner as unsigned policies, without requiring the signed Base Policy replacement procedure or UEFI deactivation sequence.

1. Disable the mechanism used to deploy the Supplemental Policy from BigFix® or your management tool to prevent automatic redeployment.
2. Proceed directly to Stage 2 for artifact removal.

Stage 2 — Policy Artifact Removal

Once Stage 1 is complete — or immediately for Option B Supplemental Policies — remove the targeted `.cip` policy file from all active policy storage locations:

- `%OSDrive%\Windows\System32\CodeIntegrity\CiPolicies\Active\{PolicyId}.cip`
- `<EFI System Partition>\Microsoft\Boot\CiPolicies\Active\{PolicyId}.cip`

After deleting the policy artifacts from both locations, restart the endpoint to complete policy removal and clear the policy state from system memory.

Operational Safety and Execution Continuity

The two-stage retirement sequence intentionally separates policy state neutralization from policy artifact removal. This design ensures continuous execution continuity for critical services — including the BigFix® Agent and core operating system processes — throughout the planned policy retirement lifecycle. Because this is an in-band workflow executed while the operating system remains fully operational, the BigFix Agent remains reachable and responsive at every stage of the procedure.

Policy Removal Tasks

Uninstall WDAC Components from Target Endpoints *(on page 75)*

Uninstall BigFix Application Control Configuration from the WDAC Management Node *(on page 73)*

Uninstall BigFix Application Control Configuration from the Root Server *(on page 70)*

Policy Management

Create and Manage WDAC Policies on the WDAC Management Node *(on page 49)*

Compile, Sign, and Publish WDAC Policies from the WDAC Management Node *(on page 60)*

Microsoft Reference

Microsoft App Control for Business — Disable App Control Policies

Chapter 8. WDAC Policy Boot-Stop Failure and Emergency Recovery

An App Control policy that blocks a boot-critical driver or core system component causes a boot-stop failure (BSOD) and prevents Windows from starting. When standard in-band recovery is not possible, administrators must use out-of-band firmware-level procedures to regain file-system access, remove problematic policy artifacts, and restore a secure boot posture.

Causes of Boot-Stop Failure

A WDAC policy that blocks a boot-critical driver or core system component prevents Windows from completing startup, resulting in a boot-stop failure (BSOD). Boot failures can also occur when a signed policy is tampered with, when policy artifacts are deleted directly without an authorized replacement, or when a signer mismatch is encountered — all of which Code Integrity treats as unauthorized modification attempts.

Important: The policy retirement procedure described in the Standard WDAC Policy Removal and Retirement Procedure topic is an **in-band, planned administrative workflow** performed while Windows and the BigFix Agent are fully operational. The emergency recovery procedures described in this topic are **out-of-band recovery workflows** used when a policy-related boot failure prevents Windows or the BigFix Agent from starting, or when a signed policy cannot be updated using an authorized signer. Do not apply out-of-band recovery steps to a functioning system undergoing planned decommission.

First-Line Automated Recovery — Option 10 (Enabled: Boot Audit On Failure)

Option 10 serves as the primary OS-level safety net for boot-critical driver blocks. If a boot-critical driver is blocked during startup, Windows automatically shifts policy enforcement to Audit Mode for that driver, allowing the operating system to complete boot processing so administrators can investigate via event logs.

Attention: Option 10 mitigates boot failures caused by blocked drivers only. It does not prevent all categories of boot-stop failure and does not override signed-policy UEFI tamper protections. For failures outside the scope of Option 10 — including signed policy removal without an authorized signer — firmware-level recovery is required.

Firmware Recovery via Secure Boot

When a policy block prevents Windows from booting entirely, disabling UEFI Secure Boot in system firmware provides the out-of-band override. Disabling Secure Boot unbinds hardware-rooted policy checks and suspends WDAC policy enforcement across both kernel and user modes, granting administrative file-system access to clear problematic policy binaries.

This approach also applies when a signed Base Policy must be retired but its original signing certificate is lost, expired, or otherwise unavailable, and a replacement policy cannot be authorized through `<UpdatePolicySigners>`.

Under these conditions, Windows Code Integrity rejects all in-band update attempts. Disabling Secure Boot bypasses the signature requirement, allowing administrators to directly delete `.cip` policy artifacts without triggering a BSOD.

BitLocker Impact During Recovery

Disabling UEFI Secure Boot alters TPM Platform Configuration Register (PCR 7) measurements that BitLocker uses to validate platform integrity. Because these integrity checks fail under a modified boot state, the TPM will refuse to automatically unseal the Volume Master Key, requiring administrators to enter the system's 48-digit BitLocker Recovery Key at startup to unlock the OS volume.

To avoid repeated recovery key prompts during multi-reboot recovery procedures, administrators should **temporarily suspend BitLocker protection** before disabling Secure Boot. Once emergency policy remediation is complete and UEFI Secure Boot has been re-enabled, BitLocker protection must be **resumed immediately** to re-bind TPM PCR 7 measurements and restore standard drive encryption posture.



Remember: Suspending BitLocker is a temporary operational step for the duration of the recovery procedure only. Failing to resume BitLocker protection after re-enabling Secure Boot leaves the drive encryption posture degraded.

Emergency Artifact Cleanup Locations

Emergency recovery requires removing the targeted `.cip` policy file from both active policy storage locations:

- `%OSDrive%\Windows\System32\CodeIntegrity\CiPolicies\Active\{PolicyId}.cip`
- `<EFI System Partition>\Microsoft\Boot\CiPolicies\Active\{PolicyId}.cip`



Warning: Because emergency removal operates out-of-band without BigFix® orchestration, policy binaries must be verified as completely removed from **both** active storage locations before restoring Secure Boot. A residual `.cip` file in either path leaves Code Integrity partially locked and will re-trigger a BSOD on the next restart.

Posture Restoration

Re-enabling Secure Boot in system firmware re-establishes the hardware-rooted security baseline. Complete all of the following steps before returning the endpoint to managed operation:

1. Verify that the `.cip` policy artifact has been removed from both active storage locations (see Emergency Artifact Cleanup Locations above).
2. Re-enable Secure Boot in system firmware.
3. If BitLocker protection was suspended during the recovery procedure, resume BitLocker protection immediately after Secure Boot is restored to re-bind TPM PCR 7 measurements and restore drive encryption posture.
4. Verify that Windows starts normally and that the BigFix Agent re-establishes contact with the Root Server.

Planned Policy Retirement

Standard WDAC Policy Removal and Retirement Procedure *(on page 26)*

Policy Management

Create and Manage WDAC Policies on the WDAC Management Node *(on page 49)*

Compile, Sign, and Publish WDAC Policies from the WDAC Management Node *(on page 60)*

Related Considerations

BitLocker Key Requirement Consideration *(on page 25)*

BigFix Application Control v2.x: Policy Behavior and Deployment Considerations *(on page 32)*

Decommission Tasks

Uninstall WDAC Components from Target Endpoints *(on page 75)*

Chapter 9. BigFix Application Control v2.x: Policy Behavior and Deployment Considerations

BigFix Application Control v2.x enforces application execution through a defined set of behavioral rules governing Base and Supplemental Policy interaction, explicit Deny rule precedence, multiple Base Policy evaluation (logical AND), script enforcement under UMCI, FilePath rule security, signed policy protection, application dependency enforcement, and signing certificate requirements. Understanding these 16 behaviors is essential before designing, deploying, or updating Application Control policies in BigFix-managed environments.

Base and Supplemental Policy Behavior

Supplemental Policies extend the trust established by their associated Base Policy and can be used to allow additional applications without modifying the Base Policy.

Base Policy Deny

An application explicitly denied by a Base Policy cannot be allowed through a Supplemental Policy. To permit the application, the Base Policy must be updated to remove the Deny and provide the required Allow rule.

Multiple Base Policies

Multiple active Base Policies are evaluated using an intersection (logical AND). An application must be allowed by all active Base Policies to execute. If any active Base Policy does not allow the application, execution is blocked.

Base Policy Updates

An existing Base Policy can be updated to change the authorization of an application. An explicit Allow or Deny can be changed by updating and redeploying the Base Policy.

Microsoft Windows Baseline

A Microsoft-provided policy template or appropriate Windows baseline should be used as the foundation for the Base Policy to help ensure required Windows components remain trusted and the operating system operates normally.

PowerShell and Script Enforcement

When UMCI is enabled and Option 11 (Disabled: Script Enforcement) is not enabled, script enforcement is active. Interactive PowerShell operates in Constrained Language Mode (CLM). Script execution depends on whether the script is trusted by the App Control policy and the applicable PowerShell configuration.

FilePath Rules

FilePath rules are subject to runtime protection based on the security of the target path. Applications located in paths that can be modified by standard users may not satisfy the default FilePath protection requirements.

Signed Policy Protection

Signed App Control policies provide protection against unauthorized modification or removal when Secure Boot is enabled. Customers should use supported policy update and removal procedures rather than manually modifying policy files.

Application Dependencies

Allowing an application's main executable may not be sufficient for the application to function. Required supporting executables, DLLs, or other components may also need to satisfy the App Control policy.

Publisher and Signer Rules

Publisher- and signer-based authorization depends on Windows Code Integrity being able to validate the application's supported signing information. Hash or file-based rules can be used when signer-based authorization is not suitable.

Policy-Signing Certificate

The certificate used to sign an App Control policy must meet the requirements of the policy-signing workflow, including the required Code Signing EKU.

Policy Signing

Signed App Control policies support RSA 2048-, 3072-, and 4096-bit keys. ECDSA is not supported for signed App Control policies.

Application Signer Rules

Application signer-based rules support RSA keys up to 4096 bits. ECC/ECDSA-based signer rules are not supported.

TPM-Based Policy Signing

A TPM-backed, non-exportable private key can be used to protect the policy-signing key. TPM availability is required when this signing approach is used.

User-Mode Enforcement

UMCI must be enabled when Application Control is intended to enforce policy on user-mode applications and scripts.

Windows Platform Support

Application Control capabilities and policy-management functionality depend on the supported Windows version, build, and edition. Policies and required features should be validated against the specific Windows platforms targeted for deployment.

DefaultWindows Policy EKU Differences

The EKU entries in the DefaultWindows policy can vary across Windows versions and builds. Customers should validate the policy contents against the specific Windows releases targeted for deployment.

Publisher Rule Size Constraint

Publisher rule generation is subject to a **4096-byte limit** for application signing information. If exceeded, a **Hash rule** may be required.

Publisher Rule – DLL Dependencies

Publisher rules for an application executable do not necessarily allow its dependent DLLs. **Unsigned DLLs may require File Path or Hash-based rules for enforcement.**

File Attribute – OriginalFilename Dependency

File Attribute rules based on `OriginalFilename` require the attribute to be present in the application binary. If missing or unavailable, **the rule may not match the application**.

Product Context

BigFix Application Control: Product Overview

System Architecture

WDAC Rule Types and Use Cases *(on page 8)*

BigFix Application Control: WDAC Known Limitations *(on page 18)*

Policy Authoring and Management

Managing WDAC Policies on the Management Node *(on page 49)*

Creating a New Base WDAC Policy *(on page 105)*

Deploying a Supplemental Policy *(on page 113)*

Certificate and Signing Setup

Provisioning the TPM-Backed Code-Signing Certificate *(on page 44)*

Compiling, Signing, and Publishing WDAC Policies *(on page 60)*

Troubleshooting and FAQs

Frequently Asked Questions *(on page 141)*

Troubleshooting Application Control *(on page 156)*

Chapter 10. Administering Application Control v2.x

This chapter covers the 11 administrator tasks required to deploy and operate BigFix Application Control v2.x using the centralized WDAC Management Node model — from Root Server installation, WDAC Management Node configuration, and TPM-backed certificate provisioning through policy authoring, compilation, signing, endpoint deployment, policy review, and full component removal.

Important: Tasks from Application Control v2.x that used the distributed model — including per-endpoint self-signed certificate creation, direct Base and Supplemental Policy deployment on individual endpoints, and related Fixlets — are deprecated. Refer to the Deprecated Application Control v2.x Fixlets and Tasks *(on page 90)* chapter for reference documentation on those tasks.

All the tasks described in this section must be performed by users with an administrator role. The tasks cover the full lifecycle of BigFix Application Control v2.x, from initial installation and configuration through policy management, deployment, and removal.

BigFix Application Control v2.x introduces a centralized WDAC Management Node model. Policy authoring, generation, signing, and publication are performed on a dedicated WDAC Management Node rather than on individual managed endpoints. BigFix Server Components handle deployment execution, endpoint management, and telemetry collection.

The following tasks are covered under this chapter:

Installation and Configuration

- **Configure and Install BigFix Application Control (BAC) on the Root Server**: Configures and installs the required BigFix Application Control components on the BigFix Root Server, including the WDAC Deployment Service, Custom Site setup, and Enterprise Database initialization. This is the first step in deploying Application Control v2.x.
- **Configure and Install the WDAC Management Node**: Provisions the dedicated WDAC Management Node by deploying the BigFix Agent, the WDAC Policy Service, and the Custom Policy Generation Engine. The WDAC Management Node is the centralized host for all WDAC policy authoring, generation, signing, and publication operations.
- **Provision the TPM-Backed Code-Signing Certificate on the WDAC Management Node**: Generates and provisions the TPM-backed RSA code-signing certificate on the WDAC Management Node. The private key is bound to the Trusted Platform Module (TPM) and never leaves it. This certificate is used to digitally sign all WDAC policies and to authenticate publication requests to the WDAC Deployment Service using TPM-signed JWTs.

Policy Management

- **Manage WDAC Policies on the WDAC Management Node**: Covers policy authoring, version assignment, rule management (allow/deny), and policy metadata persistence. Includes creating new Base Policies and Supplemental Policies, updating existing policy versions, and validating policy configuration before generation. All policy lifecycle operations are performed on the WDAC Management Node through the BigFix UI.
- **Compile, Sign, and Publish WDAC Policies from the WDAC Management Node**: Generates the compiled WDAC policy binary (`.cip`), signs it using the TPM-backed certificate, and publishes the signed artifact to the WDAC Deployment Service. The Deployment Service validates the TPM-signed JWT and publishes the policy to the BigFix Custom Site for downstream deployment. Reference binaries used during policy generation are removed from the node after compilation.

Endpoint Setup and Policy Deployment

- **Set Up and Initialize WDAC on Target Endpoints**: Prepares managed Windows endpoints for WDAC policy enforcement. This includes verifying endpoint prerequisites, checking for and handling any existing Smart App Control (SAC) policy, and initializing the endpoint to receive centrally deployed WDAC policies from the BigFix Relay.
- **Deploy and Refresh WDAC Policies on Target Endpoints**: Deploys signed Base and Supplemental Policy artifacts from the BigFix Custom Site to managed endpoints via the BigFix Relay. Applies version validation (anti-rollback) and calls RefreshPolicy to activate enforcement. Monitors deployment execution status and collects Code Integrity telemetry and compliance state via BigFix Analysis.

Policy Review

- **Submit Policy for Review**: Submits a drafted or updated WDAC policy for administrative review and approval before it is signed and published. Ensures that all WDAC policies are validated, version-controlled, and appropriately scoped prior to enterprise-wide deployment, in accordance with the policy governance controls of Application Control v2.x.

Removal and Uninstallation

- **Uninstall BigFix Application Control Configuration from the Root Server**: Removes the BigFix Application Control components from the BigFix Root Server, including the WDAC Deployment Service configuration, Custom Site content, and associated credentials from the credential store.
- **Uninstall BigFix Application Control Configuration from the WDAC Management Node**: Removes the WDAC Policy Service, Custom Policy Generation Engine, and associated configuration from the WDAC Management Node. Includes removal of the TPM-backed signing certificate and any locally cached policy artifacts.
- **Uninstall WDAC Components from Target Endpoints**: Removes all WDAC policy artifacts, associated files, and Application Control components from managed endpoints. Performs a controlled policy removal sequence to avoid endpoint disruption during uninstallation.

Prerequisites and Product Context

BigFix Application Control: Product Overview

System Architecture

BigFix Application Control v2.x: Policy Behavior and Deployment Considerations *(on page 32)*

BigFix Application Control: WDAC Known Limitations *(on page 18)*

Installation and Configuration Tasks

Configure and Install BigFix Application Control on the Root Server *(on page 37)*

Configure and Install the WDAC Management Node *(on page 41)*

Provision the TPM-Backed Code-Signing Certificate on the WDAC Management Node *(on page 44)*

Policy Management and Deployment Tasks

Manage WDAC Policies on the WDAC Management Node *(on page 49)*

Compile, Sign, and Publish WDAC Policies from the WDAC Management Node *(on page 60)*

Submit Policy for Review *(on page 79)*

Set Up and Initialize WDAC on Target Endpoints *(on page 64)*

Deploy and Refresh WDAC Policies on Target Endpoints *(on page 66)*

Removal and Uninstallation Tasks

Uninstall BigFix Application Control Configuration from the Root Server *(on page 70)*

Uninstall BigFix Application Control Configuration from the WDAC Management Node *(on page 73)*

Uninstall WDAC Components from Target Endpoints *(on page 75)*

Reference and Migration

Deprecated Application Control v2.x Fixlets and Tasks *(on page 90)*

Migrating from BigFix Application Control v1.0.0 to v2.x *(on page 97)*

Frequently Asked Questions *(on page 141)*

Troubleshooting Application Control *(on page 156)*

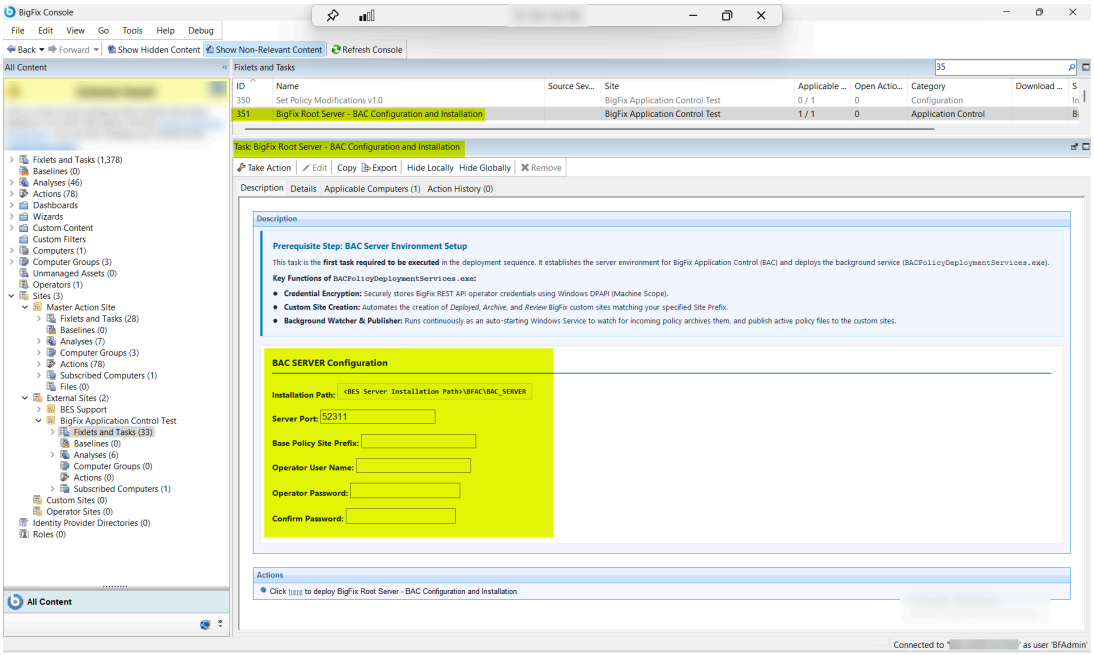
Configure and Install BigFix Application Control on the Root Server

Task ID 351 is the first required step in the BigFix Application Control deployment sequence. It installs `BACPolicyDeploymentServices.exe` on the BigFix Root Server — configuring Windows DPAPI credential encryption for the BigFix REST API, creating the Deployed, Archive, and Review policy custom sites for the specified site prefix, and starting the Background Watcher Publisher Windows Service that continuously watches for incoming policy archives and publishes active policies to the custom sites.

This is the first task (*ID 351*) required to be executed in the deployment sequence. It establishes the server environment for BigFix Application Control (BAC) and deploys the background engine *BACPolicyDeploymentServices.exe*. The key functions of *BACPolicyDeploymentServices.exe* are as follows:

- **Credential Encryption**: Securely stores **BigFix REST API** operator credentials using *Windows DPAPI* (machine scope).
- **Custom Site Creation**: Automates the creation of *Deployed*, *Archive*, and *Review* BigFix custom sites matching your specified site prefix.
- **Background Watcher Publisher**: Runs continuously as an auto-starting Windows Service to watch for incoming policy archives, version them, and publish active policy files to the custom sites.

Figure 1. BigFix Console — BAC Configuration and Installation Task (ID 351) Overview



1. In the BigFix Console, navigate to **All Content > BigFix Application Control > Fixlets and Tasks**.
2. From the **Fixlets and Tasks** pane, select **Task: BigFix Root Server - BAC Configuration and Installation**.
3. From the **Task: BigFix Root Server - BAC Configuration and Installation Description** tab, under **BAC SERVER Configuration**, enter the following information:

Figure 2. BAC Server Configuration Fields — Root Server Installation Parameters

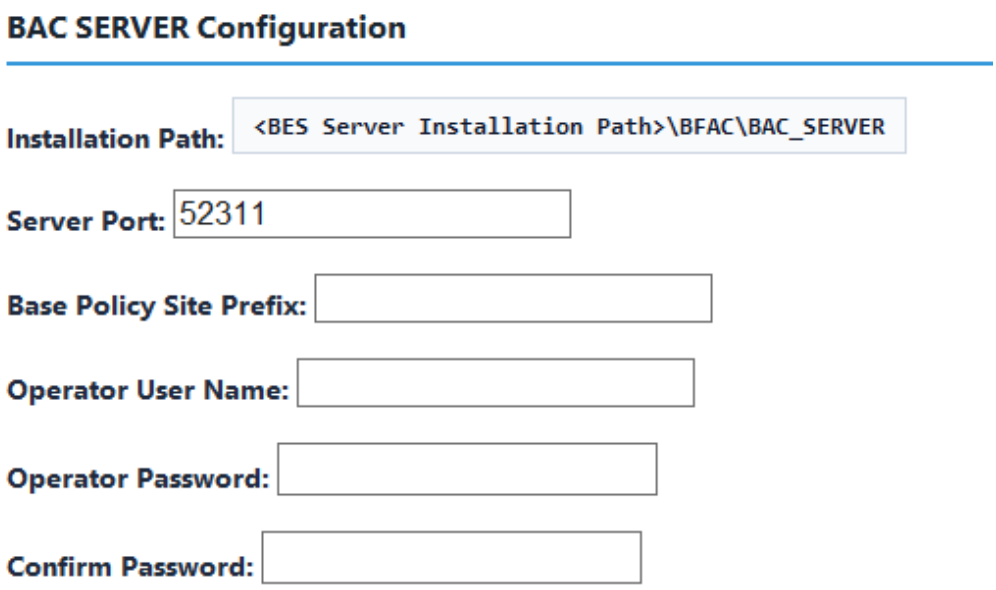


Table 3. BAC Server Configuration: Root Server Installation Parameter Descriptions

Field Name	Description
Installation Path	The local file system path on the BigFix Root Server where *BACPolicyDeploymentServices.exe* and supporting components will be deployed (e.g., `C:\Program Files\...\BAC` or `D:\Applications\BAC`). Ensure adequate disk space and that the path exists or is writable by the executing account.
Server Port	The network port used by `BACPolicyDeploymentServices` to connect to the BigFix® REST API on the Root Server. This must match the REST API port configured during BigFix Platform installation.
Base Policy Site Prefix	A user-defined namespace prefix for custom site names created during this installation. This prefix is prepended to generate site names: `<prefix>_Deployed`, `<prefix>_Archive`, and `<prefix>_Review`. This standardizes site naming across multi-environment deployments (e.g., `PROD_` generates `PROD_Deployed`, `PROD_Archive`, `PROD_Review`).
Operator User Name	The BigFix operator credentials (username) used to authenticate with the BigFix REST API. This account must hold sufficient console access to create custom sites and manage policy publication. The credentials will be securely encrypted using Windows DPAPI (machine scope) after installation.

Field Name	Description
Operator Password	The password associated with the BigFix operator account. This password is encrypted and stored securely for use by *BACPolicy-DeploymentServices* during policy archive processing and custom site publication. Do not store this password elsewhere after task completion.
Confirm Password	Verification of the operator password to prevent entry errors. Both password fields must match exactly for the installation to proceed.

4. From the **Task: BigFix Root Server - BAC Configuration and Installation** pane, click the **Applicable Computers(n)** tab and view the endpoints on which you want to run the task.
5. Select the **Take Actions** tab and select the endpoints on which you want to apply this installer task.
6. Click **OK**.

A successful execution of this task results in the following outcomes:

- **BACPolicyDeploymentServices.exe** is installed and configured at the specified path on the BigFix Root Server.
- The Background Watcher Publisher Windows Service is installed and set to auto-start on system boot.
- BigFix REST API operator credentials are securely encrypted using Windows DPAPI (machine scope) and stored for service use.
- Three custom BigFix sites are automatically created using the specified site prefix: `<prefix>_Deployed`, `<prefix>_Archive`, and `<prefix>_Review`.
- The WDAC Deployment Service is ready to receive, version, and publish policy archives to the custom sites.
- The Root Server is now ready for the next deployment step (Task ID 352: Configure and Install the WDAC Management Node).

Prerequisites and Product Context

BigFix Application Control: Product Overview

System Architecture

Administering BigFix Application Control v2.x *(on page 35)*

Next Steps in the Deployment Sequence

Configure and Install the WDAC Management Node *(on page 41)*

Provision the TPM-Backed Code-Signing Certificate on the WDAC Management Node *(on page 44)*

Manage WDAC Policies on the WDAC Management Node *(on page 49)*

Compile, Sign, and Publish WDAC Policies from the WDAC Management Node *(on page 60)*

Set Up and Initialize WDAC on Target Endpoints *(on page 64)*

Removal

Uninstall BigFix Application Control Configuration from the Root Server *(on page 70)*

Reference and Support

BigFix Application Control v2.x: Frequently Asked Questions *(on page 141)*

BigFix Application Control: Troubleshooting WDAC Enforcement Issues *(on page 156)*

Configure and Install the WDAC Management Node

Task ID 352 is the second step in the BigFix Application Control deployment sequence, requiring Task ID 351 (Root Server installation) to be completed first. It provisions the dedicated WDAC Management Node (BAC_MGMT) by creating the required folder structure (Certs, Logs, Policies, and Tools) under the BigFix Client installation path, and downloads and extracts `WdacManager.exe` — the management utility required for WDAC policy creation on the Management Node.

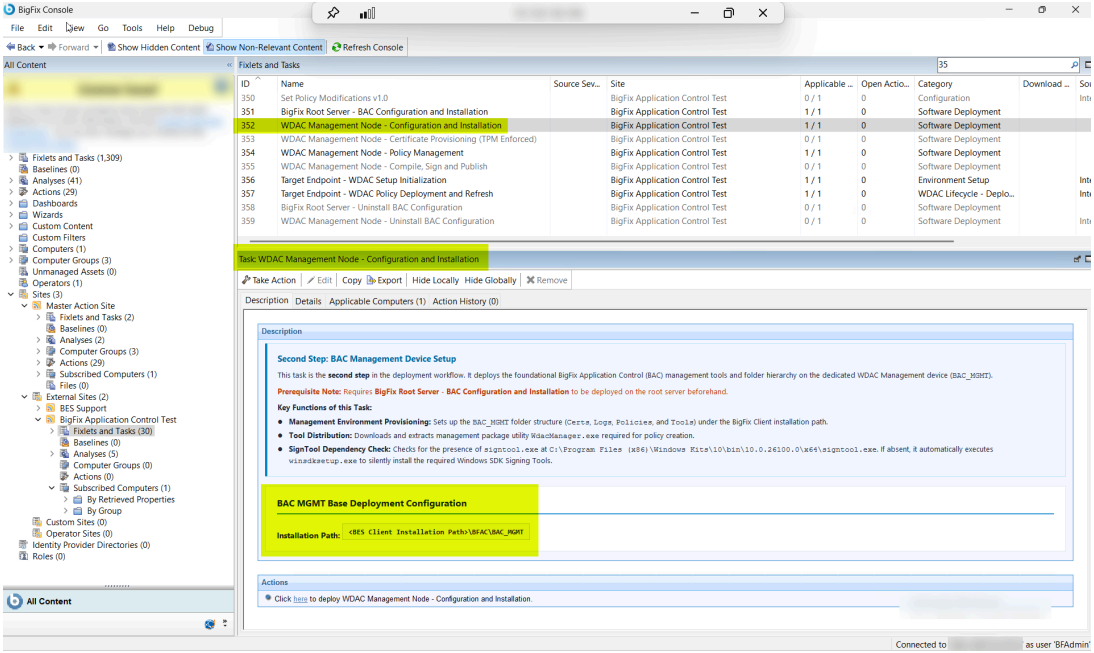
Requires Task ID 351 Configure and Install BigFix Application Control on the Root Server *(on page 37)* to be deployed on the root server beforehand.

This task (*ID 352*) is the second step in the deployment workflow. It deploys the foundational BigFix Application Control (BAC) management tools and folder hierarchy on the dedicated WDAC Management device (*BAC_MGMT*).

The task performs the following key functions:

- **Management Environment Provisioning**: Sets up the `BAC_MGMT` folder structure (*Certs*, *Logs*, *Policies*, and *Tools*) under the BigFix Client installation path.
- **Tool Distribution**: Downloads and extracts management package utility `WdacManager.exe` required for policy creation.
- **SignTool Dependency Check:** Checks for the presence of `signtool.exe` at `C:\Program Files (x86)\Windows Kits\10\bin\10.0.26100.0\x64\signtool.exe`. If absent, it automatically executes `winsdksetup.exe` to silently install the required Windows SDK Signing Tools.

Figure 3. BigFix Console — WDAC Management Node Configuration and Installation Task (ID 352) Overview



1. In the BigFix Console, navigate to **All Content > BigFix Application Control > Fixlets and Tasks**.
2. From the **Fixlets and Tasks** pane, select **WDAC Management Node - Configuration and Installation**.
3. From the **Task: WDAC Management Node - Configuration and Installation** pane, under **BAC MGMT Base Deployment Configuration** select one of the following policy options:

Figure 4. WDAC Management Base Deployment Configuration Options — Installation Path

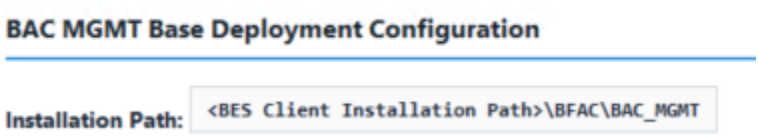


Table 4. WDAC Management Node Configuration Options: Installation Parameter Descriptions

Field Name	Description
Installation Path	The local file system path on the WDAC Management Node where the BAC_MGMT folder structure will be created under the BigFix Client installation directory (e.g., `C:\Program Files (x86)\BigFix Enterprise\BES Client\BAC_MGMT`). This path must exist and be writable by the executing account. The four required subfolders (*Certs*, *Logs*, *Policies*, and *Tools*) will be automatically created here during installation.

4. From the **Task: WDAC Management Node - Configuration and Installation options** pane, click the **Applicable Computers(n)** tab and view the endpoints on which you want to run the task.
5. Select the **Take Actions** tab and select the endpoints on which you want to apply this installer task.
6. Click **OK**.

A successful execution of this task results in the following outcomes:

- The BAC_MGMT folder structure is created under the BigFix Client installation path with four required subfolders: *Certs* (for code-signing certificates), *Logs* (for task logs and audit trails), *Policies* (for WDAC policy source files), and *Tools* (for management utilities).
- WdacManager.exe (the management package utility for WDAC policy creation) is downloaded and extracted to the `Tools` subfolder, ready for policy authoring.
- The Windows SDK Signing Tools (**signtool.exe**) are verified; if absent, **winsdksetup.exe** is automatically executed to silently install the required signing tools at `C:\Program Files (x86)\Windows Kits\10\bin\10.0.26100.0\x64\`.
- The WDAC Management Node is now fully provisioned and ready for TPM-backed code-signing certificate generation (Task ID 353).

Prerequisites and Product Context

Configure and Install BigFix Application Control on the Root Server *(on page 37)*

System Architecture

Administering BigFix Application Control v2.x *(on page 35)*

Next Steps in the Deployment Sequence

Provision the TPM-Backed Code-Signing Certificate on the WDAC Management Node *(on page 44)*

Manage WDAC Policies on the WDAC Management Node *(on page 49)*

Compile, Sign, and Publish WDAC Policies from the WDAC Management Node *(on page 60)*

Set Up and Initialize WDAC on Target Endpoints *(on page 64)*

Removal

Uninstall BigFix Application Control Configuration from the WDAC Management Node *(on page 73)*

Reference and Context

BigFix Application Control v2.x: Policy Behavior and Deployment Considerations *(on page 32)*

BigFix Application Control v2.x: Frequently Asked Questions *(on page 141)*

BigFix Application Control: Troubleshooting WDAC Enforcement Issues *(on page 156)*

Provision the TPM-Enforced WDAC Policy Signing Certificate on the WDAC Management Node

Provision a TPM 2.0–enforced code-signing certificate on the dedicated WDAC Management Node (BAC_MGMT) to enable secure WDAC policy signing. Choose from three certificate modes: self-signed using the Microsoft Platform Crypto Provider, Enterprise CA integrated with Active Directory PKI, or a customer-managed OpenSSL CA using a two-pass CSR generation and trust store import workflow. This task requires Task 352 (WDAC Management Node installation) to be completed first.

Requires Task 352 (Configure and Install the WDAC Management Node *(on page 41)*) to be completed on the management target beforehand.

This task (*ID 353*) is the third step in the deployment workflow. It provisions the WDAC policy signing certificate on the dedicated WDAC Management device (BAC_MGMT).

The task has the following Certificate Mode options:

- **Self Signed**: Generates a TPM hardware-bound self-signed certificate locally using *Microsoft Platform Crypto Provider*. Custom validity (1-5 years) is supported. Outputs both binary `.cer` and Base64 `.pem` files.
- **CA External (OpenSSL / Customer-Managed)**: Operates in a two-stage sequential workflow using **Run 1** and **Run 2**.

Detailed Operational Flow for Option 2 (CA External / OpenSSL):

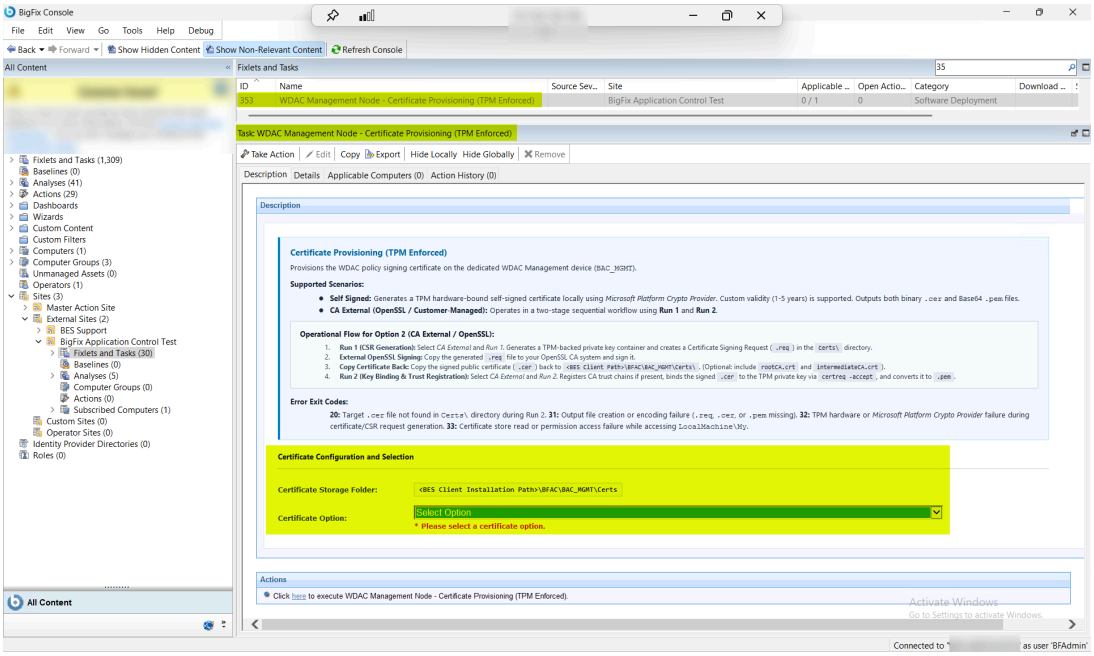
1. **Run 1 (CSR Generation)**: Select *CA External* and *Run 1*. Generates a TPM-backed private key container and creates a *Certificate Signing Request (.req)* in the `Certs\` directory.
2. **External OpenSSL Signing**: Copy the generated *.req* file to your OpenSSL CA system and sign it.
3. **Copy Certificate Back**: Copy the signed public certificate (*.cer*) back to `<BES Client Path>\BFAC\BAC_MGMT\Certs\`. (Optional: include *rootCA.crt* and *intermediateCA.crt*).
4. **Run 2 (Key Binding & Trust Registration):** Select *CA External* and *Run 2*. Registers *CA trust chains* if present, binds the signed *.cer* to the TPM private key via certreq -accept, and converts it to *.pem*.

Refer to the table below to know more about the task's exit code.

Table 5. Task Exit Code Reference: Certificate Provisioning Failure Codes

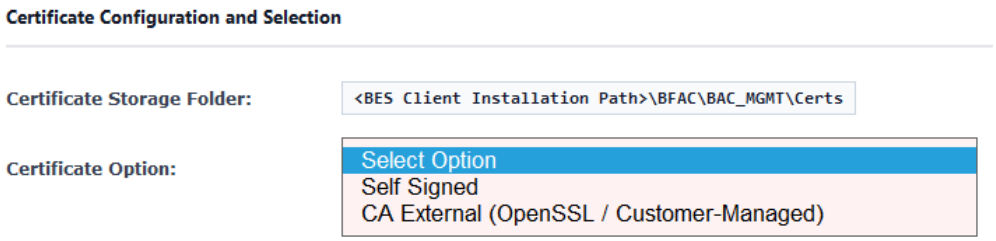
Exit Code	Description
20	Target **.cer** file not found in the `Certs\` directory during **Run 2**.
31	Output file creation or encoding failure (**.req**, **.cer**, or **.pem** missing).
32	TPM hardware or *Microsoft Platform Crypto Provider* failure during certificate/CSR request generation.
33	Certificate store read or permission access failure while accessing `LocalMachine\My`.

Figure 5. BigFix Console — WDAC Management Node Certificate Provisioning Task (ID 353) Overview



1. In the BigFix Console, navigate to **All Content > BigFix Application Control > Fixlets and Tasks**.
2. From the **Fixlets and Tasks** pane, select **Task: WDAC Management Node - Certificate Provisioning (TPM ENFORCED)**.
3. From the **Task: WDAC Management Node - Certificate Provisioning (TPM ENFORCED)** pane, under **Certificate Configuration and Selection**, select the desired certificate option - **Self Signed** or **CA External (OpenSSL / Customer-Managed)**.

Figure 6. WDAC Management Node Certificate Provisioning — Configuration Options: Certificate Mode Selection



a. For the **Self Signed** option, fill the following fields:

Figure 7. WDAC Management Node Certificate Provisioning — Self Signed Certificate Option

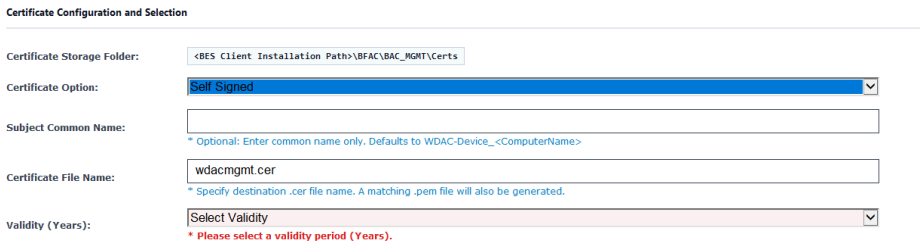


Table 6. Certificate Option — Self Signed: Field Descriptions

Field Name	Description
Certificate Storage Folder	The directory on the WDAC Management Node where the generated certificate files (`.cer` and `.pem`) are written. Defaults to `<BES Client Path>\BFAC\BAC_MGMT\Certs\`. Ensure this path exists and is writable by the BigFix Client service account before running the task.
Certificate Option	Set to **Self Signed** to generate a TPM hardware-bound self-signed certificate locally using the Microsoft Platform Crypto Provider. Select this mode when no enterprise CA or external certificate authority is available. The private key is generated inside the TPM and cannot be exported.
Subject Common Name	The X.509 Subject Common Name (CN) for the self-signed certificate. This value appears in the signature metadata of all WDAC policies signed by `WdacManager.exe`. Use a name that identifies the WDAC Management Node or organization, for example `WDAC_Management_Node_Prod` or `BAC_MGMT_Policy_Signing`. Maximum 64 characters; avoid special characters except hyphens and underscores.
Certificate File Name	The base filename for the generated certificate output files. The task writes two files to the Certificate Storage Folder using this name: a binary DER-encoded file (`.cer`) and a Base64-encoded PEM file (`.pem`). For example, entering `wdac_self_signed` produces `wdac_self_signed.cer` and `wdac_self_signed.pem`.
Validity (Years)	The certificate lifetime in years. Supported range: 1 to 5 years. Choose based on your organization's certificate lifecycle policy. A shorter validity period (1–2 years) increases security through more frequent rotation; a longer period (3–5 years) reduces operational

Field Name	Description
	overhead but requires proactive renewal planning before expiration to prevent policy signing outages.

b. For the **CA External (OpenSSL / Customer-Managed)** option, fill the following fields:

Figure 8. WDAC Management Node Certificate Provisioning — CA External (OpenSSL / Customer-Managed) Certificate Option

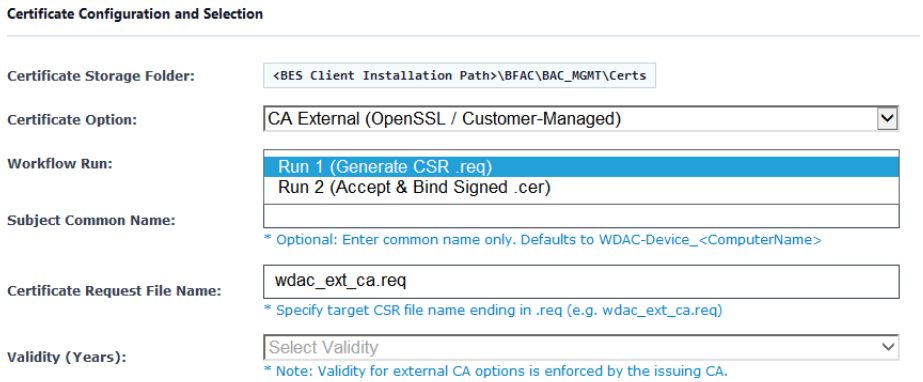


Table 7. Certificate Option — CA External (OpenSSL / Customer-Managed): Field Descriptions

Field Name	Description
Certificate Storage Folder	The directory on the WDAC Management Node used for both the outgoing CSR file (Run 1) and the incoming signed certificate and CA chain files (Run 2). Defaults to `<BES Client Path>\BFAC\BAC_MGMT\Certs\`. Before running Run 2, ensure the signed certificate (`.cer`) and any CA chain files (`rootCA.crt`, `intermediate-CA.crt`) are copied to this folder.
Certificate Option	Set to **CA External (OpenSSL / Customer-Managed)** to use the two-pass CSR workflow. Select this mode when your organization uses a customer-managed or third-party certificate authority (such as an OpenSSL CA, Sectigo, or DigiCert) that requires an external certificate signing request.
Workflow Run	Identifies which pass of the two-pass workflow to execute: ▪ **Run 1**: Creates the TPM-backed private key container and generates the Certificate Signing Request (`.req`) file in the Certificate Storage Folder. Run this first, then transmit the `.req` file to your external CA for signing. ▪ **Run 2**: Imports the signed certificate and CA chain files from the Certificate Storage Folder, binds the certificate to the TPM private key container via certreq -machine -ac-

Field Name	Description
	cept, and registers CA chain certificates in the Windows trust store. Run this after the signed certificate is returned by your external CA and copied to the Certificate Storage Folder.
Subject Common Name	The X.509 Subject Common Name (CN) to embed in the Certificate Signing Request. This must match the CN your external CA will sign. Use a name that identifies the WDAC Management Node or organization, for example `WDAC_Management_Node_Prod` or `BAC_MGMT_-Policy_Signing`. The CN appears in all signed WDAC policies and must match exactly across Run 1 and Run 2.
Certificate Request File Name	The filename for the Certificate Signing Request output file generated in Run 1. The task writes a `.req` file with this name to the Certificate Storage Folder. Default: `wdac_ext_ca.req`. After Run 1, copy this file to your external CA system for signing. The signed certificate returned by your CA (typically `wdacmgmt_ext_ca.cer`) is imported during Run 2.
Validity (Years)	The requested certificate lifetime in years (1 to 5). This value is included in the CSR and submitted to your external CA; the actual validity period in the signed certificate is determined by your CA's policy. If your CA overrides the requested validity, ensure the issued certificate validity is acceptable before proceeding to Run 2.

4. From the **Task: WDAC Management Node - Certificate Provisioning (TPM ENFORCED)** pane, click the **Applicable Computers(n)** tab and view the endpoints on which you want to run the task.
5. Select the **Take Actions** tab and select the endpoints on which you want to apply this installer task.
6. Click **OK**.

A successful execution provisions a TPM 2.0–enforced code-signing certificate on the WDAC Management Node. The outcome depends on the selected certificate mode:

- **Self Signed mode**: A self-signed certificate is generated using the Microsoft Platform Crypto Provider and bound to the TPM hardware module. Binary (`.cer`) and Base64 (`.pem`) certificate files are written to the Certificate Storage Folder. No external CA is required; the certificate is immediately available for WDAC policy signing by `WdacManager.exe`.
- **CA External — Run 1 (CSR Generation)**: A TPM-backed private key container is created and a Certificate Signing Request (`.req`) is written to the Certificate Storage Folder. Transmit the `.req` file to your external CA for signing, then copy the signed certificate (and any CA chain files) back to the Certificate Storage Folder before proceeding to Run 2.

- **CA External — Run 2 (Key Binding and Trust Registration)**: The signed certificate is bound to the TPM private key container via certreq -machine -accept. Root and intermediate CA chain certificates present in the Certificate Storage Folder are registered in the Windows trust store (`Cert:\LocalMachine\Root` and `Cert:\LocalMachine\CA`). The certificate is converted to `.pem` format for use by `WdacManager.exe`.

After successful provisioning, `WdacManager.exe` can use the TPM-bound signing certificate to sign WDAC policies. Verify the binding by opening **Certificates (Local Computer)** > **Personal** > **Certificates** and confirming the Subject CN and Key Storage Provider (Microsoft Platform Crypto Provider) match your configuration. The WDAC Management Node is now ready for the next step in the deployment sequence: managing and compiling WDAC policies.

Tip: If the task fails with error `TRUST_E_CERT_SIGNATURE (0x80096004)`, verify that all root and intermediate CA certificates are imported into `Cert:\LocalMachine\Root` and `Cert:\LocalMachine\CA` before running the task. See the troubleshooting guide for resolution steps.

Prerequisites and Product Context

Configure and Install the WDAC Management Node *(on page 41)*

Configure and Install BigFix Application Control on the Root Server *(on page 37)*

Administering BigFix Application Control v2.x *(on page 35)*

System Architecture

Next Steps in the Deployment Sequence

Manage WDAC Policies on the WDAC Management Node *(on page 49)*

Compile, Sign, and Publish WDAC Policies from the WDAC Management Node *(on page 60)*

Set Up and Initialize WDAC on Target Endpoints *(on page 64)*

Removal

Uninstall BigFix Application Control Configuration from the WDAC Management Node *(on page 73)*

Reference and Troubleshooting

BigFix Application Control v2.x: Policy Behavior and Deployment Considerations *(on page 32)*

BigFix Application Control v2.x: Frequently Asked Questions *(on page 141)*

BigFix Application Control: Troubleshooting WDAC Enforcement Issues *(on page 156)*

Create and Manage WDAC Policies on the WDAC Management Node

Create and manage WDAC Base and Supplemental policy XML files on the dedicated WDAC Management Node (BAC_MGMT) directly from the BigFix® Console. The Create Policies workflow guides you through three sequential steps — configuration inputs, rule options, and custom parameter rules. Use the Purge Supplemental Policy Rules

workflow to remove specific rules from existing Supplemental policies, and the Add / Remove Signer workflow to inject or strip code-signing certificates from Base or Supplemental policy XML files.

Requires Task 353 (Provision the TPM-Enforced WDAC Policy Signing Certificate on the WDAC Management Node *(on page 44)*) to be completed on the BAC_MGMT Management Node beforehand. The TPM-bound signing certificate must be provisioned before policies can be authored and signed.

This task (*ID 354*) interactively generates WDAC Base/Supplemental policies or purge rules from existing Supplemental Policies directly from the BigFix Console.

- **Create Policies**: Build new XML Base or Supplemental policies from templates, configure rule options, and define custom rule parameters (**Path**, **Hash**, **Publisher**, **COM**, **Package**, **File Attributes**).
- **Purge Supplemental Policy Rules**: Remove specific rule ID values or hashes from an existing Supplemental Policy XML and output a cleaned, updated policy XML file.
- **Add / Remove Signer**: Strip specified Signer IDs from existing Base or Supplemental policies and inject update or supplemental signing certificates (**.cer**) for secure policy deployment.

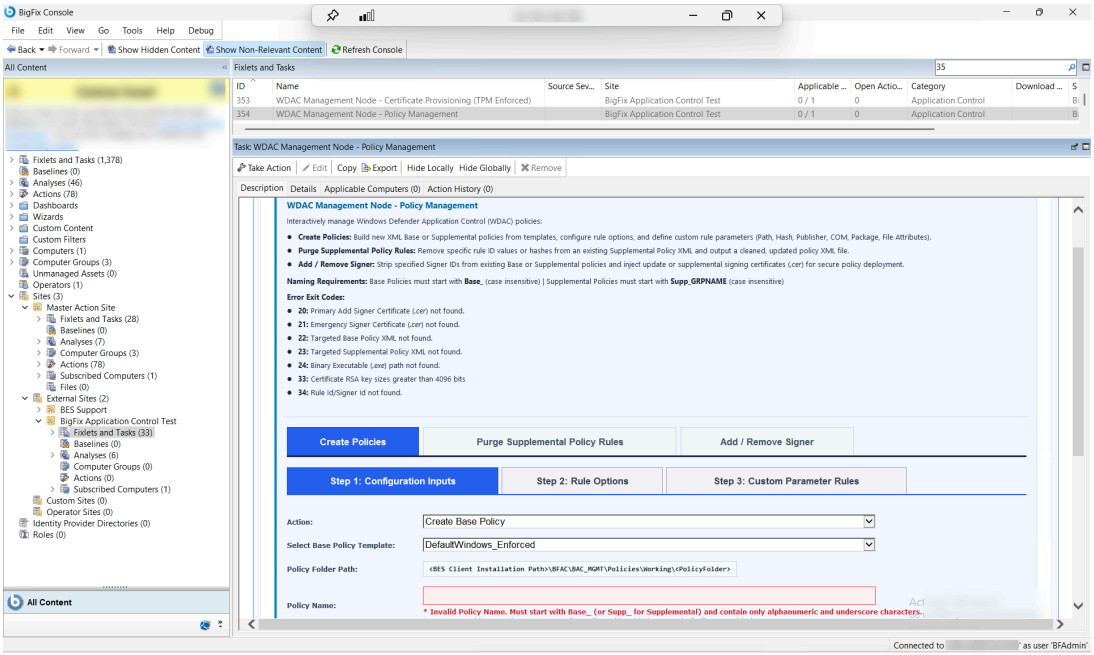
Naming Requirements: Base policies must start with `Base_` | Supplemental Policies must start with `Supp_GRPNAME`.

Refer to the table below to know more about the task's exit code.

Table 8. Task Exit Code Reference: Policy Management Failure Codes

Exit Code	Description
20	Primary Add Signer Certificate (**.cer**) not found.
21	Emergency Signer Certificate (**.cer**) not found.
22	Targeted Base Policy XML not found.
23	Targeted Supplemental Policy XML not found.
24	Binary Executable (**.exe**) path not found.
33	Certificate RSA key sizes greater than 4096 bits.
34	Rule Id/Signer Id not found.

Figure 9. BigFix Console — WDAC Management Node Policy Management Task (ID 354) Overview



1. In the BigFix Console, navigate to **All Content > BigFix Application Control > Fixlets and Tasks**.
2. From the **Fixlets and Tasks** pane, select **WDAC Management Node - Policy Management**.
3. From the **WDAC Management Node - Policy Management** pane, under **Create Policies**, enter the following.

    a. Navigate to the first tab **Create Policies > Step 1: Configuration Inputs** to configure its options.

Figure 10. Create Policies — Step 1: Configuration Inputs

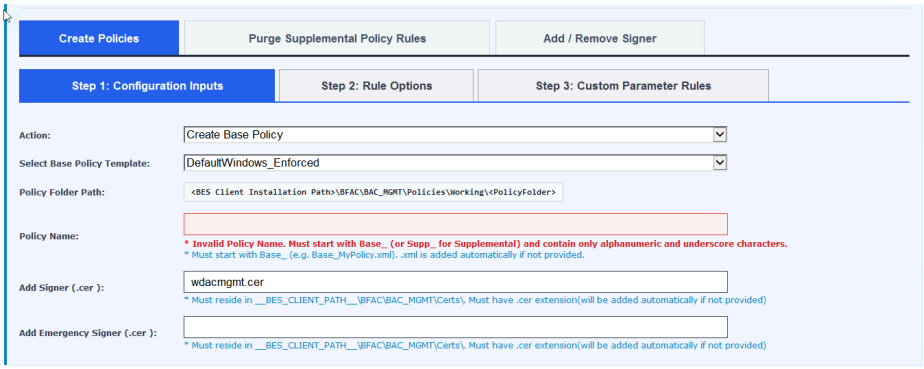


Table 9. Create Policies Step 1: Configuration Inputs — Field Descriptions

Field Name	Description
Action	Selects the policy creation workflow: **Create Base Policy** or **Create Supplemental Policy**. Base policies define the primary trust boundary for a set of endpoints; Supplemental policies extend a Base policy with group-specific allow

Field Name	Description
	or deny rules. The selection determines which naming requirements and template options apply.
Select Base Policy Template	The Microsoft-provided WDAC policy template to use as the starting point. Templates define the initial set of trusted signers and default rule options. Common options include Default Windows (allows Windows-signed binaries only), Allow Microsoft (allows Microsoft-signed and WHQL-signed binaries), and Windows Audit (broad trust in audit mode). Choose based on the intended enforcement level for the target endpoints.
Policy Folder Path	The directory on the WDAC Management Node where the generated policy XML file is written. Defaults to the `BAC_MGMT\Policies\` directory. Ensure the path exists and is writable by the BigFix Client service account before running the task.
Policy Name	The name of the output policy XML file, without the `.xml` extension. Naming is enforced: Base policy names must start with `Base_` (for example, `Base_Production`); Supplemental policy names must start with `Supp_GRPNAME`, where `GRPNAME` is the target endpoint group identifier (for example, `Supp_FinanceGroup`). The task rejects names that do not conform to these conventions.
Add Signer (.cer)	The full path to the primary code-signing certificate (`.cer` file) on the WDAC Management Node — typically the TPM-bound signing certificate provisioned in Task 353. The certificate is added as a trusted Signer in the generated policy, authorizing files signed by this certificate to run on target endpoints.
Add Emergency Signer (.cer)	The full path to an optional backup or emergency code-signing certificate (`.cer` file). Adding an emergency signer provides a fallback signing authority if the primary certificate becomes unavailable, preventing a policy-signing outage. Leave blank if no emergency signer is required.

b. Navigate to the second tab **Create Policies > Step 2: Rule Options** to select the appropriate check box options for your plan.

Figure 11. Create Policies — Step 2: Rule Options

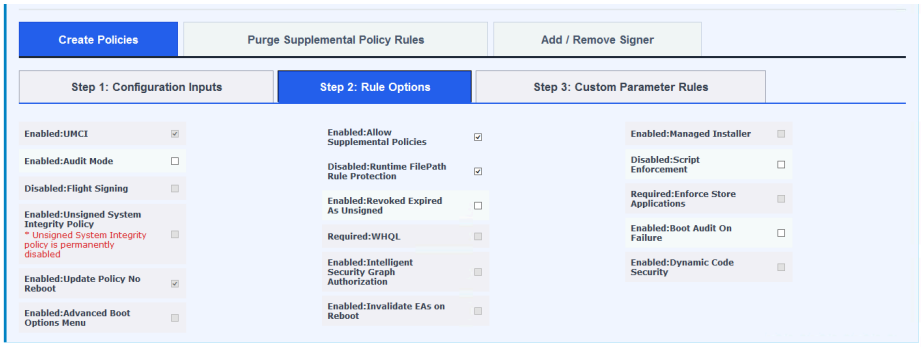


Table 10. Create Policies Step 2: Rule Options — WDAC Policy Rule Flag Descriptions

Checkbox Name	Description
Enabled: UMCI	Enables User Mode Code Integrity enforcement. When selected, WDAC policy rules apply to all user-mode processes, not only kernel drivers. Required for full application control on managed Windows endpoints.
Enabled: Audit Mode	Runs the policy in monitoring mode rather than enforcement mode. Policy violations are logged to the Windows Event Log (event IDs 3076/3077) but code execution is not blocked. Select this option to baseline a new policy on endpoints before switching to enforcement.
Disabled: Flight Signing	This option is **disabled by default** in Application Control v2.x because it requires Microsoft™ platform validation.
Enabled: Unsigned System Integrity Policy	Allows this policy XML to be deployed without a cryptographic signature. When cleared, the policy must be signed with a trusted certificate before deployment. Not recommended for production environments — clear this option and sign all policies for security.
Enabled: Update Policy No Reboot	Allows policy updates to take effect at the next refresh cycle without requiring a system reboot. When cleared, policy changes require a reboot to take effect. Select for environments where minimizing downtime during policy updates is a priority.
Enabled: Advanced Boot Menu Options	This option is **disabled by default** in Application Control v2.x because it requires Microsoft™ platform validation.
Enabled: Allow Supplemental Policies	Allows this Base policy to be extended by Supplemental policies deployed from the BigFix Custom Site. Must be selected on any Base policy that will be paired with

Checkbox Name	Description
	group-specific Supplemental policies. Has no effect on Supplemental policies themselves.
Disabled: Runtime FilePath Rule Protection	Disables Windows protection against runtime bypass of file path rules (for example, via symbolic link or junction attacks). Select only when path-based rules are required and the reduced security of unprotected path rules is an accepted trade-off.
Enabled: Revoked Expired as Unsigned	Treats binaries signed with revoked or expired certificates as unsigned rather than categorically denying them. When selected, such binaries are evaluated against unsigned-code rules rather than being blocked outright.
Required: WHQL	This option is **disabled by default** in Application Control v2.x as this option is kernel-based but BigFix Application Control v2.x works in user mode.
Enabled: Intelligent Security Graph Authorization	This option is **disabled by default** in Application Control v2.x as this option is kernel-based but BigFix Application Control v2.x works in user mode.
Enabled: Managed Installer	This option is **disabled by default** in Application Control v2.x as this option is kernel-based but BigFix Application Control v2.x works in user mode.
Disabled: Script Enforcement	Disables WDAC enforcement for scripting hosts (PowerShell, VBScript, JScript, WSH). When selected, scripts run without policy checks. Select during initial deployment phases where script-heavy workloads are not yet covered by policy rules; clear in production to enforce full code integrity.
Required: Enforce Store Applications	This option is **disabled by default** in Application Control v2.x because it requires Microsoft™ platform validation.
Enabled: Boot Audit On Failure	Switches the policy to audit mode at boot if the policy fails to load normally, rather than blocking the system. Prevents boot failures caused by policy load errors. Recommended for initial policy deployments on production endpoints.
Enabled: Dynamic Code Security	This option is **disabled by default** in Application Control v2.x because it requires Microsoft™ platform validation

Checkbox Name	Description
	and also as this option is kernel-based but BigFix Application Control v2.x works in user-mode.

To know more about the **Rule Options**, refer to Understand App Control for Business policy rules and file rules.

c. Navigate to the third tab **Create Policies > Step 3: Custom Parameter Rules** to configure its options.

Figure 12. Create Policies — Step 3: Custom Parameter Rules

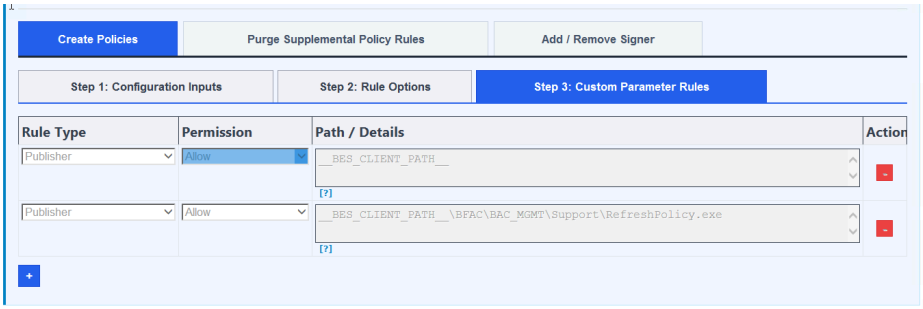


Table 11. Create Policies Step 3: Custom Parameter Rules — Field Descriptions

Field Name	Description
Rule Type	The type of custom rule to add to the policy: **Path** (allow or deny by file system path, with wildcard support), **Publisher** (allow or deny by code-signing certificate subject name, covering all versions from that publisher), or **File Hash** (allow or deny by SHA-256 cryptographic hash of a specific file version). Hash rules are the most precise but require updates when file content changes; Publisher rules are broader but automatically cover future versions from the same signer.
Permission	Whether the rule grants or blocks execution: **Allow** (the specified file, path, or publisher is trusted and permitted to run) or **Deny** (explicitly blocked, regardless of other allow rules). Deny rules take precedence over Allow rules in WDAC policy evaluation.
Path Details	The input value for the selected rule type: a file system path for Path rules (for example, `C:\Program Files\MyApp\*`), a publisher certificate subject name for Publisher rules, or a SHA-256 hash value for File Hash rules. For Path rules, environment variable substitutions and wildcards (`*`) are supported. For Hash rules, use Get-AppLockerFileInformation or Get-SystemDriver to obtain the correct hash value.

4. From the **WDAC Management Node - Policy Management** pane, under **Purge Supplemental Policy Rules**, enter the following.

Figure 13. Purge Supplemental Policy Rules — Configuration Fields Overview

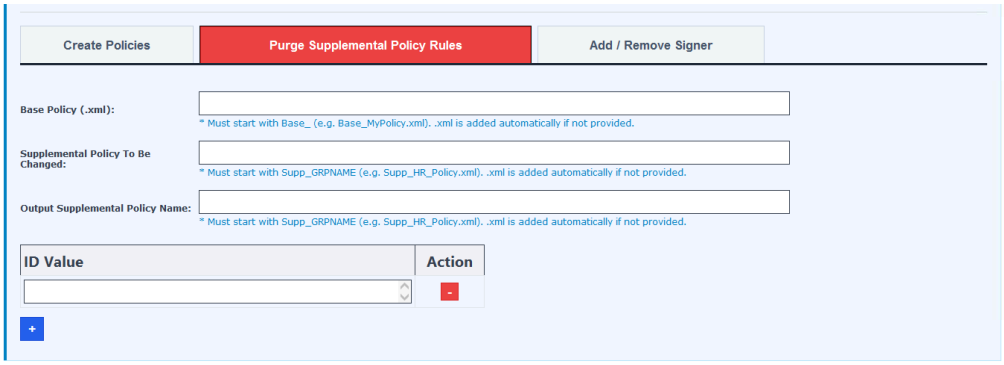
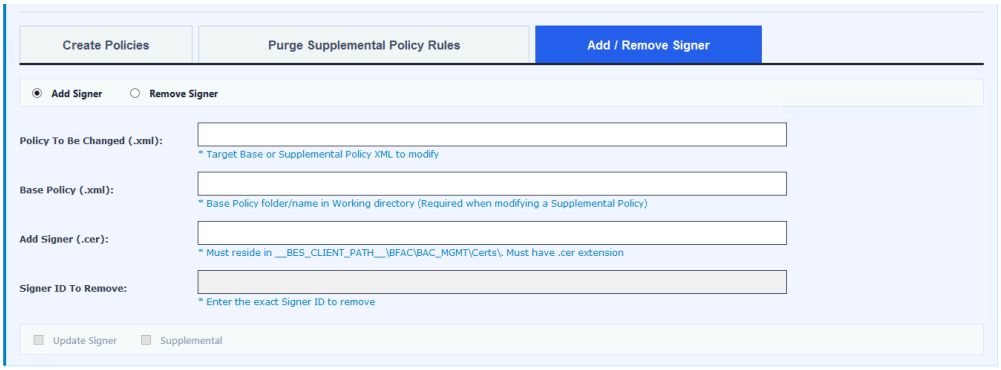


Table 12. Purge Supplemental Policy Rules — Field Descriptions

Field Name	Description
Base Policy (.xml)	The full path to the Base Policy XML file associated with the Supplemental policy being purged. The task uses the Base Policy to validate that the purged output remains a valid child policy. The Base Policy file must already exist in the Policy Folder Path on the WDAC Management Node.
Supplemental Policy To Be Changed	The file name (or full path) of the existing Supplemental Policy XML from which specific rules are to be removed. The task reads this file, removes the specified rule IDs or hashes, and writes the cleaned policy to a new output file. The original Supplemental Policy XML is not modified.
Output Supplemental Policy Name	The name for the new, cleaned Supplemental Policy XML output file, without the `.xml` extension. Must follow the `Supp_GRPNAME` naming convention. The task writes the purged policy to this filename in the Policy Folder Path on the WDAC Management Node.

5. From the **WDAC Management Node - Policy Management** pane, under **Add / Remove Signer**, select either the **Add Signer** or **Remove Signer** radio option.

Figure 14. Add / Remove Signer — Action Selection and Configuration Overview



a. For the **Add Signer** option, enter the following options.

Table 13. Add Signer — Policy and Certificate Field Descriptions

Field Name	Description
Policy To Be Changed (.xml)	The full path to the Base or Supplemental Policy XML file into which the signing certificate will be injected. The task adds the specified certificate as a new trusted Signer element in the policy XML. The updated policy must be re-compiled and re-published to apply the change to target endpoints.
Base Policy (.xml)	The full path to the parent Base Policy XML file. Required when the target policy is a Supplemental policy, to validate that the new signer is consistent with the Base Policy's trust chain. When modifying a Base policy directly, this field references the Base policy being changed.
Add Signer (.cer)	The full path to the signing certificate (`.cer` file) to inject into the target policy. The certificate's subject name and public key are added as a new Signer element, authorizing files signed by this certificate to run on endpoints where the policy is deployed.

b. For the **Remove Signer** option, enter the following options.

Figure 15. Add / Remove Signer — Remove Signer Configuration Fields



Table 14. Remove Signer — Policy and Signer ID Field Descriptions

Field Name	Description
Policy To Be Changed (.xml)	The full path to the Base or Supplemental Policy XML file from which the specified Signer ID will be stripped. The task removes the matching Signer element from the policy XML. The updated policy must be re-compiled and re-published to revoke trust from the removed signer on target endpoints.
Base Policy (.xml)	The full path to the parent Base Policy XML file. Used to validate that removing the signer does not break the policy's trust chain. Required when the target policy is a Supplemental policy.
Signer ID To Remove	The Signer ID value to strip from the target policy XML, for example `ID_SIGNER_A_1`. Signer IDs are found in the policy XML under the `<Signers>` element. Use a text editor or run Get-CIPolicy -FilePath <policy.xml> in PowerShell to identify the correct Signer ID before running this operation.

6. From the **WDAC Management Node - Policy Management** pane, click the **Applicable Computers(n)** tab and view the endpoints on which you want to run the task.
7. Select the **Take Actions** tab and select the endpoints on which you want to apply this installer task.
8. Click **OK**.

The outcome of a successful execution depends on the selected workflow:

- **Create Base Policy**: A new WDAC Base Policy XML file is written to the Policy Folder Path using the selected template, the configured rule options, and any custom parameter rules. The file is named with the `Base_` prefix and is ready for compilation, signing, and publication in the next deployment step.
- **Create Supplemental Policy**: A new WDAC Supplemental Policy XML file is written to the Policy Folder Path, linked to the parent Base Policy. The file is named with the `Supp_GRPNAME` prefix and contains the group-specific allow or deny rules defined in Step 3.
- **Purge Supplemental Policy Rules**: A new, cleaned Supplemental Policy XML is written using the Output Supplemental Policy Name. The specified rule IDs or hashes are removed from the output policy. The original Supplemental Policy XML is not modified.
- **Add Signer**: The specified signing certificate is injected into the target policy XML as a new trusted Signer element. The updated policy trusts files signed by the added certificate. Re-compile and re-publish the policy to apply the change to target endpoints.
- **Remove Signer**: The specified Signer ID is stripped from the target policy XML. The updated policy no longer trusts files signed by the removed signer. Re-compile and re-publish the policy to apply the change to target endpoints.

All output policy XML files are written to the Policy Folder Path on the WDAC Management Node. Proceed to **Compile, Sign, and Publish WDAC Policies from the WDAC Management Node** to compile, sign with the TPM certificate, and publish the updated policies to the BigFix Custom Site for endpoint deployment.

Prerequisites and Deployment Context

Provision the TPM-Enforced WDAC Policy Signing Certificate on the WDAC Management Node *(on page 44)*

Configure and Install the WDAC Management Node *(on page 41)*

Administering BigFix Application Control v2.x *(on page 35)*

Next Steps in the Deployment Sequence

Compile, Sign, and Publish WDAC Policies from the WDAC Management Node *(on page 60)*

Set Up and Initialize WDAC on Target Endpoints *(on page 64)*

Deploy and Refresh WDAC Policies on Target Endpoints *(on page 66)*

Removal

Remove WDAC Base and Supplemental Policies *(on page 126)*

Uninstall BigFix Application Control Configuration from the WDAC Management Node *(on page 73)*

Reference and Troubleshooting

BigFix Application Control v2.x: Policy Behavior and Deployment Considerations *(on page 32)*

WDAC Limitations *(on page 18)*

BigFix Application Control v2.x: Frequently Asked Questions *(on page 141)*

BigFix Application Control: Troubleshooting WDAC Enforcement Issues *(on page 156)*

Compile, Sign, and Publish WDAC Policies from the WDAC Management Node

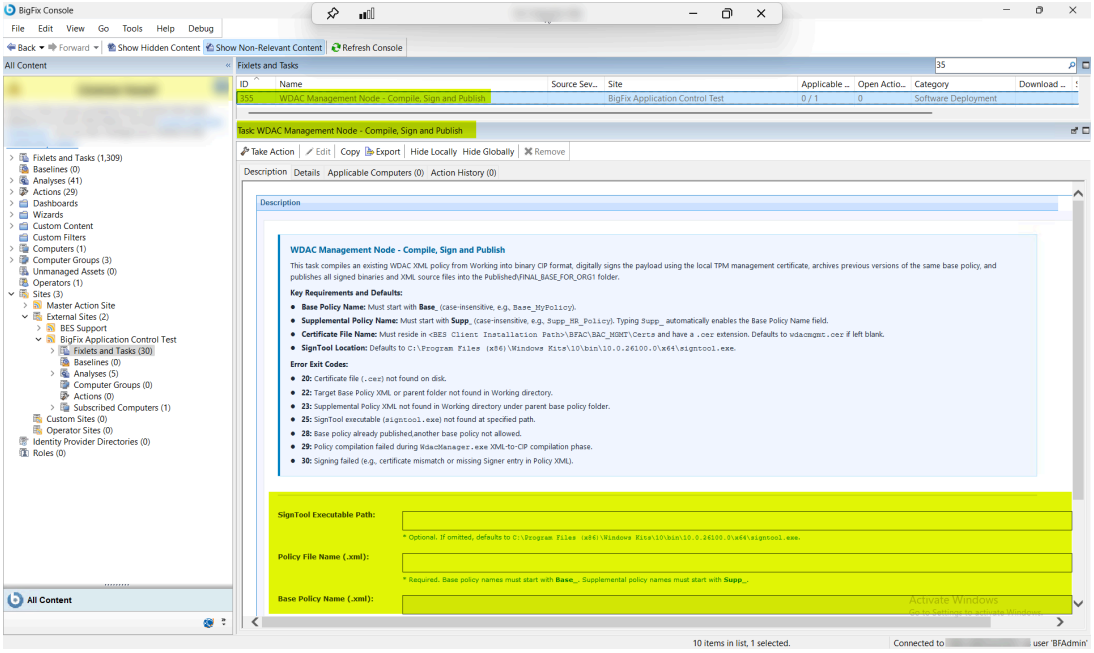
Compile an existing WDAC XML policy on the BAC_MGMT Management Node into binary CIP format using WdacManager.exe, digitally sign the compiled payload using the local TPM-bound certificate (`wdacmgmt.cer` by default), archive any previous version of the same Base policy, and publish all signed binaries and XML source files to the BigFix Custom Site for server-wide endpoint deployment. Base policy file names must start with `Base_`; Supplemental policy file names must start with `Supp_`; certificate files must use the `.cer` extension.

This task (*ID 355*) compiles an existing WDAC XML policy into binary CIP format, digitally signs the payload using the local TPM management certificate, archives previous versions of the same Base policy, and publishes all signed binaries and XML source files into the `Published\FINAL_BASE_FOR_ORG1` folder.

The task has the following key requirements and defaults:

- **Base Policy Name**: Must start with `Base_` (case-insensitive, for example, **Base_MyPolicy**).
- **Supplemental Policy Name**: Must start with `Supp_` (case-insensitive, for example, **Supp_HR_Policy**).
- **Certificate File Name**: Must reside in `<BES Client Installation Path>\BFAC\BAC_MGMT\Certs` and have a `.cer` extension. Defaults to `wdacmgmt.cer` if left blank.
- **SignTool Location**: Defaults to `C:\Program Files (x86)\Windows Kits\10\bin\10.0.26100.0\x64\signtool.exe`.

Figure 16. BigFix Console — WDAC Management Node Compile, Sign, and Publish Task (ID 355) Overview



Refer to the following exit code table for troubleshooting information if the task fails:

Table 15. Task ID 355 Exit Code Reference: Compile, Sign, and Publish Failure Codes

Exit Code	Description
20	Certificate file (**.cer**) not found on disk.
22	Target Base Policy XML or parent folder not found in Working directory.
23	Supplemental Policy XML not found in Working directory under parent base policy folder.
25	SignTool executable (**signtool.exe**) not found at specified path.
28	Base policy already published,another base policy not allowed.
29	Policy compilation failed during **WdacManager.exe** XML-to-CIP compilation phase.
30	Signing failed (e.g., certificate mismatch or missing Signer entry in Policy XML).

1. In the BigFix® Console, navigate to **All Content > BigFix Application Control > Fixlets and Tasks**.
2. From the **Fixlets and Tasks** pane, select **Task: WDAC Management Node - Compile, Sign and Publish**.
3. From the **Task: WDAC Management Node - Compile, Sign and Publish** pane, enter the following.

Figure 17. WDAC Compile, Sign, and Publish — Configuration Options

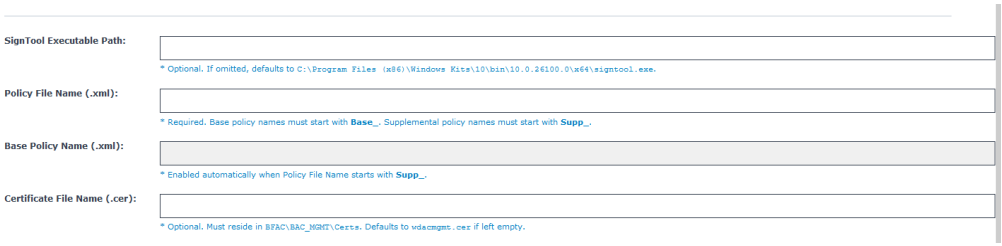


Table 16. Compile, Sign, and Publish — Configuration Options Field Descriptions

Field Name	Description
SignTool Executable Path	Full path to `signtool.exe` on the WDAC Management Node. Defaults to `C:\Program Files (x86)\Windows Kits\10\bin\10.0.26100.0\x64\signtool.exe` if left blank. Override this field if Windows Kits is installed to a non-default path or a different SDK version is installed.
Base Policy Name (.xml)	The filename of the parent Base Policy XML in the BAC_MGMT working directory, without the `.xml` extension. Required when **Is Supplemental Policy** is **Yes**. The name must start with `Base_` (case-insensitive). Leave blank when compiling a Base policy directly.
Policy File Name (.xml)	The filename of the WDAC XML policy to compile and sign, without the `.xml` extension. For Base policies, the name must start with `Base_`; for Supplemental policies, it must start with `Supp_`. The file must exist in the BAC_MGMT working directory before running the task.
Certificate File Name (.cer)	The filename of the TPM-bound signing certificate in the `BAC_MGMT\Certs` directory, including the `.cer` extension. Defaults to `wdacmgmt.cer` if left blank. The certificate must have been provisioned by Task 353 before this task can run successfully.

4. From the **Task: WDAC Management Node - Compile, Sign and Publish** pane, click the **Applicable Computers(n)** tab and view the endpoints on which you want to run the task.
5. Select the **Take Actions** tab and select the endpoints on which you want to apply this installer task.
6. Click **OK**.

A successful execution of this task produces the following outcomes on the WDAC Management Node:

- **Binary CIP file compiled**: The specified WDAC XML policy is converted to a binary `.cip` file in the BAC_MGMT working directory by `WdacManager.exe`. The CIP file is ready for signing and distribution.
- **Policy digitally signed**: The compiled CIP binary is signed with the TPM-bound management certificate (`wdacmgmt.cer` by default) using `signtool.exe`. The signed binary carries a verifiable digital signature that endpoints validate before applying the policy.
- **Previous Base policy version archived**: If a previously published version of the same Base policy exists in the `Published\FINAL_BASE_FOR_ORG1` folder, it is moved to an archive subfolder before the new version is written. This preserves rollback capability without overwriting prior builds.
- **Signed binaries and XML source published**: The signed CIP binary and the original XML source file are copied to the `Published\FINAL_BASE_FOR_ORG1` folder and uploaded to the BigFix Custom Site, making the policy available for server-wide deployment to managed endpoints.
- **Deployment ready**: The policy archive is now available in the BigFix Custom Site. Proceed to **Set Up and Initialize WDAC on Target Endpoints** (step 6) to begin endpoint-level WDAC deployment using the published policy.

Important: If the task exits with a non-zero code, refer to the exit code table in the task description to diagnose the failure before re-running the task. Exit code 28 (base policy already published) requires archiving or removing the existing published Base policy before a new Base policy can be published.

Prerequisites and Deployment Context

Create and Manage WDAC Policies on the WDAC Management Node *(on page 49)*

Provision the TPM-Enforced WDAC Policy Signing Certificate on the WDAC Management Node *(on page 44)*

Configure and Install the WDAC Management Node *(on page 41)*

BitLocker Recovery Key Requirements Before WDAC Recovery *(on page 25)*

Administering BigFix Application Control v2.x *(on page 35)*

Next Steps in the Deployment Sequence

Set Up and Initialize WDAC on Target Endpoints *(on page 64)*

Deploy and Refresh WDAC Policies on Target Endpoints *(on page 66)*

Removal

Remove WDAC Base and Supplemental Policies *(on page 126)*

Uninstall BigFix Application Control Configuration from the WDAC Management Node *(on page 73)*

Reference and Troubleshooting

BigFix Application Control v2.x: Policy Behavior and Deployment Considerations *(on page 32)*

BigFix Application Control v2.x: Frequently Asked Questions *(on page 141)*

BigFix Application Control: Troubleshooting WDAC Enforcement Issues *(on page 156)*

Set Up and Initialize WDAC on Target Endpoints

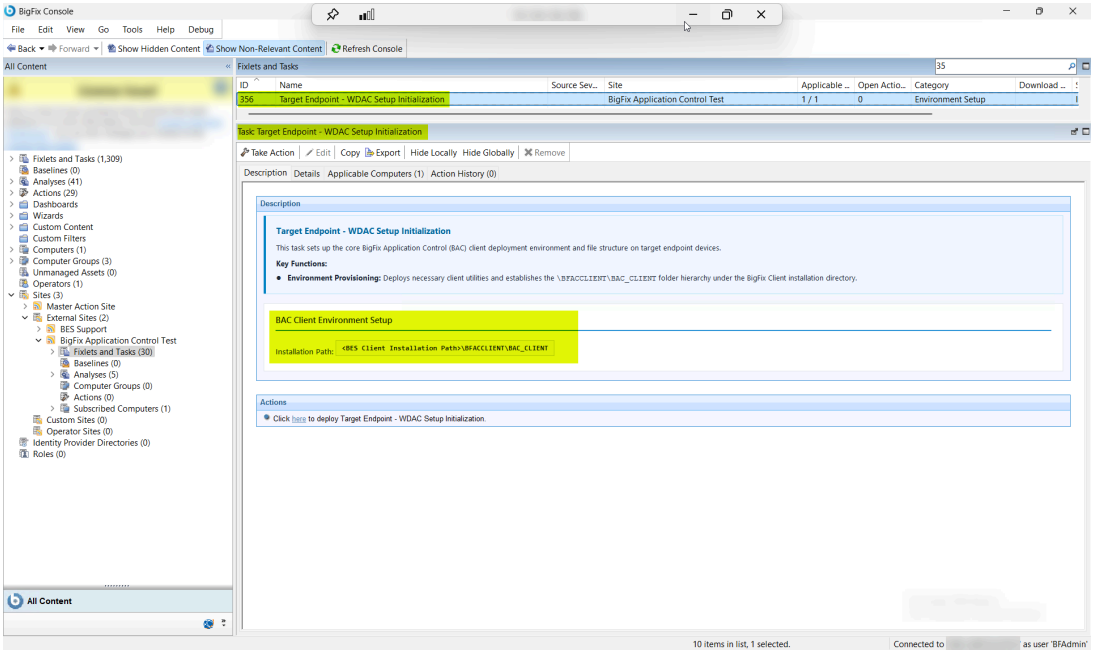
Provision the BAC client deployment environment on target Windows endpoints by deploying the required client utilities and establishing the `\BFACCLIENT\BAC_CLIENT` folder hierarchy under the BigFix Client installation directory. Specify the **Installation Path** to match the BigFix Client installation directory on target devices. Completing this task prepares target endpoints to receive, apply, and enforce WDAC policies distributed through the BigFix Custom Site.

This task (*ID 356*) sets up the core BigFix Application Control (BAC) client deployment environment and file structure on target endpoint devices.

The task has the following key function:

- **Environment Provisioning**: Deploys necessary client utilities and establishes the `\BFACCLIENT\BAC_CLIENT` folder hierarchy under the BigFix Client installation directory.

Figure 18. BigFix Console — Target Endpoint WDAC Setup Initialization Task (ID 356) Overview



1. In the BigFix Console, navigate to **All Content > BigFix Application Control > Fixlets and Tasks**.
2. From the **Fixlets and Tasks** pane, select **Task: Target Endpoint - WDAC Setup Initialization**.

3. From the **Task: Target Endpoint - WDAC Setup Initialization** pane, under **BAC Client Environment Setup**, enter the following.

Figure 19. Target Endpoint WDAC Setup Initialization — Configuration Options: Installation Path

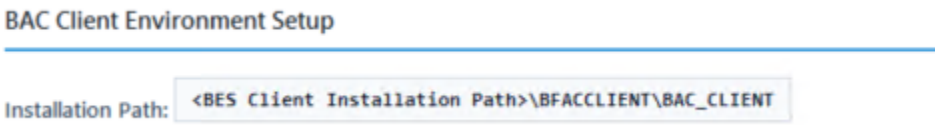


Table 17. Target Endpoint Setup — Configuration Options Field Descriptions

Field Name	Description
Installation Path	The full path to the BigFix Client installation directory on target endpoints — the root under which the task creates the `\BFAC-CLIENT\BAC_CLIENT` folder hierarchy and deploys client utilities. Defaults to the standard BigFix Client installation path (`C:\Program Files (x86)\BigFix Enterprise\BES Client`) if left blank. Override this field only if the BigFix Client was installed to a non-default path on the target endpoints.

4. From the **Task: Target Endpoint - WDAC Setup Initialization** pane, click the **Applicable Computers(n)** tab and view the endpoints on which you want to run the task.
5. Select the **Take Actions** tab and select the endpoints on which you want to apply this installer task.
6. Click **OK**.

A successful execution of this task produces the following outcomes on each target endpoint:

- **BAC_CLIENT folder hierarchy created**: The `\BFACCLIENT\BAC_CLIENT` folder structure is created under the BigFix Client installation directory on the target endpoint, establishing the on-device working directory for all subsequent WDAC policy operations.
- **Client utilities deployed**: The required BigFix Application Control client utilities are extracted and placed in the `BAC_CLIENT` directory, enabling the endpoint to receive, apply, and refresh WDAC policies distributed through the BigFix Custom Site.
- **Endpoint ready for policy deployment**: The target endpoint is now prepared to receive its first WDAC policy. Proceed to **Deploy and Refresh WDAC Policies on Target Endpoints** (Task ID 357, step 7) to push the compiled and signed Base policy from the BigFix Custom Site to initialized endpoints.

Tip: Run this task against all target endpoints before beginning policy deployment. Endpoints that have not completed this initialization step will not have the `BAC_CLIENT` folder structure in place and will fail to apply policies distributed by Task ID 357.

Prerequisites and Deployment Context

Compile, Sign, and Publish WDAC Policies from the WDAC Management Node *(on page 60)*

Create and Manage WDAC Policies on the WDAC Management Node *(on page 49)*

Provision the TPM-Enforced WDAC Policy Signing Certificate on the WDAC Management Node *(on page 44)*

Administering BigFix Application Control v2.x *(on page 35)*

System Architecture

Next Steps in the Deployment Sequence

Deploy and Refresh WDAC Policies on Target Endpoints *(on page 66)*

Submit Policy for Review *(on page 79)*

Removal

Uninstall WDAC Components from Target Endpoints *(on page 75)*

Reference and Troubleshooting

BigFix Application Control v2.x: Policy Behavior and Deployment Considerations *(on page 32)*

WDAC Limitations *(on page 18)*

BigFix Application Control v2.x: Frequently Asked Questions *(on page 141)*

BigFix Application Control: Troubleshooting WDAC Enforcement Issues *(on page 156)*

Deploy and Refresh WDAC Policies on Target Endpoints

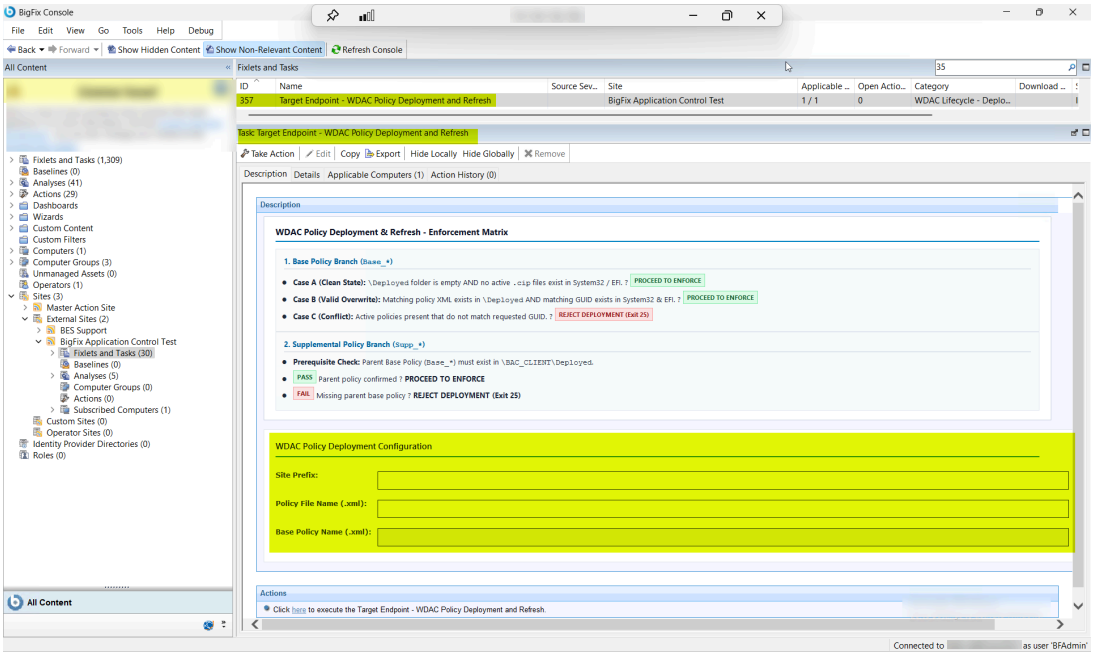
Fetch signed `.cip.p7` WDAC policy files from the BigFix Custom Site, parse policy GUIDs from file manifests, stage binary policies into the Active CodeIntegrity directory (and EFI partition for UEFI systems), and trigger `RefreshPolicy.exe` to enforce the new policies immediately — without requiring a system reboot. Before running, ensure the target endpoint has been initialized by Task ID 356 and that the policy has been compiled and published by Task ID 355. For Supplemental policies, the parent Base policy must already be present in the `\BAC_CLIENT\Deployed` folder on each target endpoint.

This task (*ID 357*) performs the deployment, GUID parsing, active staging, and live enforcement refresh of compiled WDAC binary policies on target endpoint machines.

The task applies the following enforcement matrix when determining whether a policy can be deployed:

- **Base Policy Branch (Base_*)**
  - **Case A (Clean State)**: `\Deployed` folder is empty AND no active **.cip** files exist in `System32/EFI`. Application Control v2.x can proceed to enforce it.
  - **Case B (Valid Overwrite)**: Matching policy XML exists in `\Deployed` AND matching GUID exists in `System32 & EFI`. Application Control v2.x can proceed to enforce it.
  - **Case C (Conflict)**: Active policies present that do not match requested GUID. Application Control v2.x will reject deployment with exit code 25.
- **Supplemental Policy Branch (Supp_*)**
  - **Prerequisite Check**: Parent Base Policy (Base_*) must exist in `\BAC_CLIENT\Deployed` folder.
  - If the PASS Parent policy is confirmed then Application Control v2.x can proceed to enforce it.
  - If the PASS Missing Parent base policy is confirmed then Application Control v2.x will reject deployment with exit code 25.

Figure 20. BigFix Console — Target Endpoint WDAC Policy Deployment and Refresh Task (ID 357) Overview



1. In the BigFix® Console, navigate to **All Content > BigFix Application Control > Fixlets and Tasks**.
2. From the **Fixlets and Tasks** pane, select **Task: Target Endpoint - WDAC Policy Deployment and Refresh**.
3. From the **Task: Target Endpoint - WDAC Policy Deployment and Refresh** pane, under **WDAC Policy Deployment Configuration**, enter the following.

Figure 21. WDAC Policy Deployment and Refresh — Configuration Options

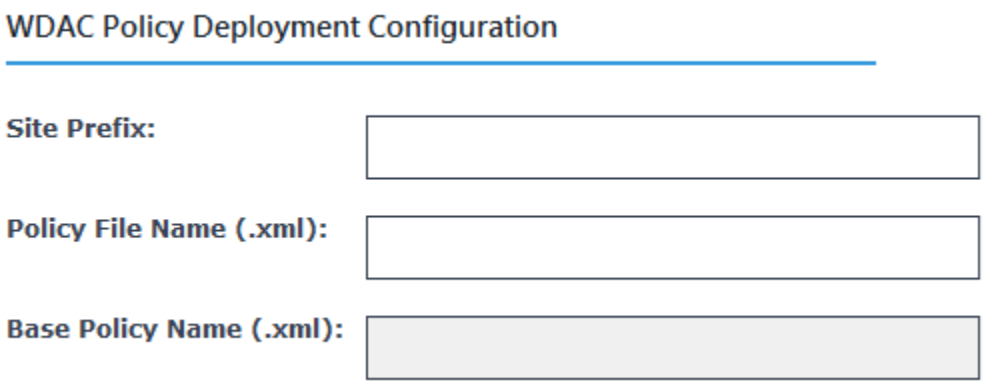


Table 18. WDAC Policy Deployment and Refresh — Configuration Options Field Descriptions

Field Name	Description
Site Prefix	The name prefix of the BigFix Custom Site from which signed `.cip.p7` policy archives are fetched. The task constructs the full Custom Site name as `CustomSite_<SitePrefix>_Policies_Deployed`. Enter the site prefix exactly as configured when the BigFix Application Control Custom Site was set up — the value is case-sensitive.
Policy File Name (.xml)	The filename of the WDAC policy to deploy, without the `.xml` extension. Must match a policy that was compiled, signed, and published to the BigFix Custom Site by Task 355. For Base policies the name must start with `Base_`; for Supplemental policies it must start with `Supp_`. The task uses this name to locate the corresponding `.cip.p7` signed binary in the Custom Site.
Base Policy Name (.xml)	The filename of the parent Base policy, without the `.xml` extension. Required only when deploying a Supplemental policy — the task verifies that the specified Base policy already exists in the `\BAC_CLIENT\Deployed` folder on the target endpoint before proceeding. Leave blank when deploying a Base policy directly.

4. From the **Task: Target Endpoint - WDAC Policy Deployment and Refresh** pane, click the **Applicable Computers(n)** tab and view the endpoints on which you want to run the task.
5. Select the **Take Actions** tab and select the endpoints on which you want to apply on this task.
6. Click **OK**.

A successful execution of this task produces the following outcomes on each target endpoint:

- **Policy archive fetched from Custom Site**: The signed `.cip.p7` policy archive is downloaded from the `CustomSite_<SitePrefix>_Policies_Deployed` Custom Site to the `\BFACCLIENT\BAC_CLIENT\Temp` staging directory on the target endpoint.
- **Policy GUID parsed and enforcement matrix evaluated**: The policy GUID is extracted from the file manifest and validated against the enforcement matrix — confirming either a clean-state deployment (Case A), a valid same-GUID overwrite (Case B), or detecting a conflicting GUID (Case C, exit code 25). For Supplemental policies, the parent Base policy prerequisite in `\BAC_CLIENT\Deployed` is verified before proceeding.
- **Binary policy staged to Active CodeIntegrity**: The compiled `.cip` binary is copied to `System32\CodeIntegrity\CiPolicies\Active`. On UEFI systems, the binary is also written to the EFI partition (`EFI\Microsoft\Boot\CiPolicies\Active`) to ensure enforcement persists across Secure Boot.
- **Policy record written to Deployed folder**: The policy XML and metadata are recorded in the `\BAC_CLIENT\Deployed` folder, enabling future Supplemental policy deployments to validate the parent Base policy prerequisite.
- **Live enforcement activated without reboot**: `RefreshPolicy.exe` is invoked to signal the Windows kernel to load and apply the newly staged policy immediately. WDAC enforcement begins on the target endpoint without requiring a system reboot.

Important: If the task exits with code 25, either a conflicting Base policy GUID is already active on the endpoint (Case C — remove the existing Base policy before deploying a new one) or the required parent Base policy is missing from `\BAC_CLIENT\Deployed` (Supplemental policy prerequisite check failed — deploy the parent Base policy first using this task, then re-run for the Supplemental policy).

Prerequisites and Deployment Context

Set Up and Initialize WDAC on Target Endpoints *(on page 64)*

Compile, Sign, and Publish WDAC Policies from the WDAC Management Node *(on page 60)*

Create and Manage WDAC Policies on the WDAC Management Node *(on page 49)*

BitLocker Recovery Key Requirements Before WDAC Recovery *(on page 25)*

Administering BigFix Application Control v2.x *(on page 35)*

Next Steps and Related Workflows

Submit Policy for Review *(on page 79)*

Remove WDAC Base and Supplemental Policies *(on page 126)*

Removal

Uninstall WDAC Components from Target Endpoints *(on page 75)*

Reference and Troubleshooting

BigFix Application Control v2.x: Policy Behavior and Deployment Considerations *(on page 32)*

WDAC Limitations *(on page 18)*

BigFix Application Control v2.x: Frequently Asked Questions *(on page 141)*

BigFix Application Control: Troubleshooting WDAC Enforcement Issues *(on page 156)*

Uninstall BigFix Application Control Configuration from the Root Server

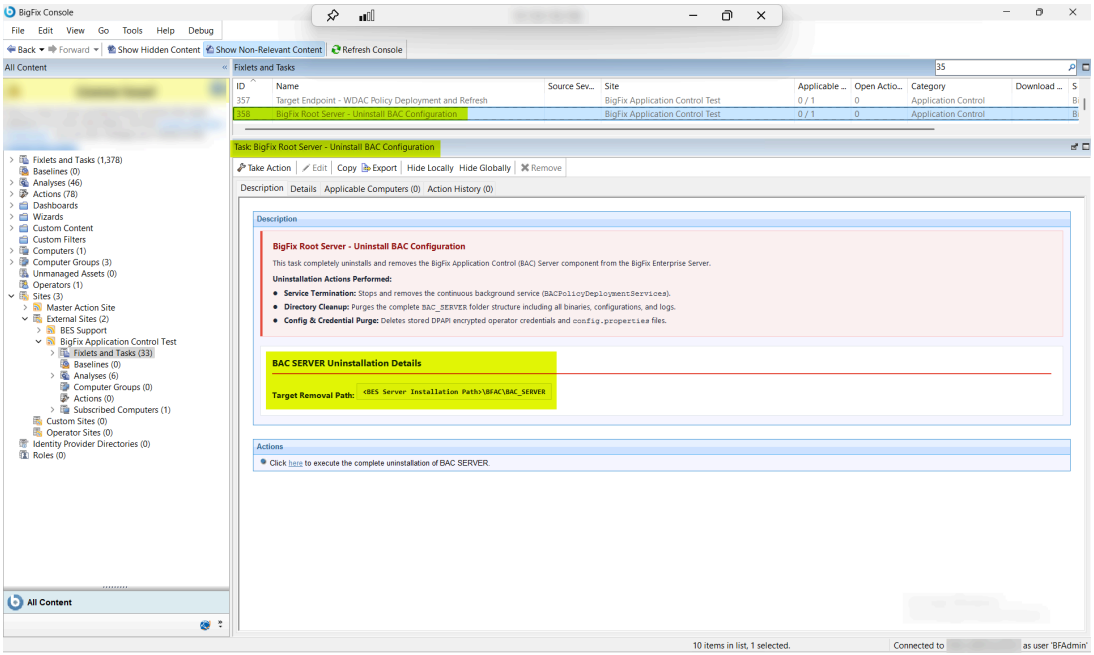
Uninstall the BigFix Application Control (BAC) Server component from the BigFix Enterprise Root Server by stopping and removing the `BACPolicyDeploymentServices` background Windows service, deleting the `BAC_SERVER` folder structure and associated executables, and purging DPAPI-encrypted operator credentials and `config.properties` files. Use the **Target Path Removal** field to specify the BAC_SERVER installation path.

This task (*ID 358*) completely uninstalls and removes the BigFix Application Control (BAC) Server component from the BigFix Enterprise Server.

The task performs the following uninstall activities:

- **Service Termination**: Stops and removes the continuous background service (`BACPolicyDeploymentServices`).
- **Directory Cleanup:** Purges the complete `BAC_SERVER` folder structure including all binaries, configurations, and logs.
- **Config & Credential Purge**: Deletes stored DPAPI encrypted operator credentials and `config.properties` files.

Figure 22. BigFix Console — Root Server Uninstall BAC Configuration Task (ID 358) Overview



1. In the BigFix® Console, navigate to **All Content > BigFix Application Control > Fixlets and Tasks**.
2. From the **Fixlets and Tasks** pane, select **Task: BigFix Root Server - Uninstall BAC Configuration**.
3. From the **Task: BigFix Root Server - Uninstall BAC Configuration** pane, under **BAC SERVER Installation Details**, enter the following.

Figure 23. Root Server BAC Uninstall — Configuration Options: Target Path Removal and Cleanup Mode

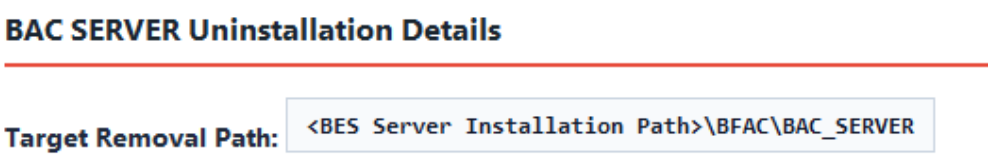


Table 19. Root Server BAC Uninstall — Configuration Options Field Descriptions

Field Name	Description
Target Path Removal	The full installation path of the BAC_SERVER folder on the BigFix Enterprise Root Server — the directory specified when Task ID 351 (Root Server BAC configuration) was installed. The task removes the BAC_SERVER folder structure, all executables, DPAPI-encrypted credential files, `config.properties`, and the `BACPolicyDeploymentServices` service from this location. Defaults to the standard BigFix Enterprise Server installation path if left blank.

4. From the **Task: BigFix Root Server - Uninstall BAC Configuration** pane, click the **Applicable Computers(n)** tab and view the endpoints on which you want to run the task.
5. Select the **Take Actions** tab and select the endpoints on which you want to apply this installer task.
6. Click **OK**.

A successful execution of this task produces the following outcomes on the BigFix Enterprise Root Server:

- **BACPolicyDeploymentServices stopped and removed**: The continuous background Windows service (`BACPolicyDeploymentServices`) is stopped and unregistered from the Windows Service Control Manager. The service no longer runs on the Root Server and does not restart on reboot.
- **BAC_SERVER folder structure removed**: The complete `BAC_SERVER` directory — including all executables, binaries, and configuration files — is deleted from the path specified in **Target Path Removal**.
- **DPAPI credentials and config.properties purged**: All DPAPI-encrypted operator credentials and `config.properties` files stored by the BAC Server component are permanently deleted. These credentials cannot be recovered after this step — document any required operator credentials before running the task.



Warning: This task permanently deletes DPAPI-encrypted operator credentials. Ensure all required credentials are documented before running this task. After completion, target endpoints remain in their last enforced WDAC policy state until their BAC_CLIENT components are also removed via **Uninstall WDAC Components from Target Endpoints**.

Installation Counterpart

Configure and Install BigFix Application Control on the Root Server *(on page 37)*

Related Uninstall Tasks

Uninstall WDAC Components from Target Endpoints *(on page 75)*

Uninstall BigFix Application Control Configuration from the WDAC Management Node *(on page 73)*

Remove WDAC Base and Supplemental Policies *(on page 126)*

Deployment Context

Administering BigFix Application Control v2.x *(on page 35)*

System Architecture

BigFix Application Control: Product Overview

Reference and Troubleshooting

BitLocker Recovery Key Requirements Before WDAC Recovery *(on page 25)*

BigFix Application Control v2.x: Frequently Asked Questions *(on page 141)*

BigFix Application Control: Troubleshooting WDAC Enforcement Issues *(on page 156)*

Uninstall BigFix Application Control Configuration from the WDAC Management Node

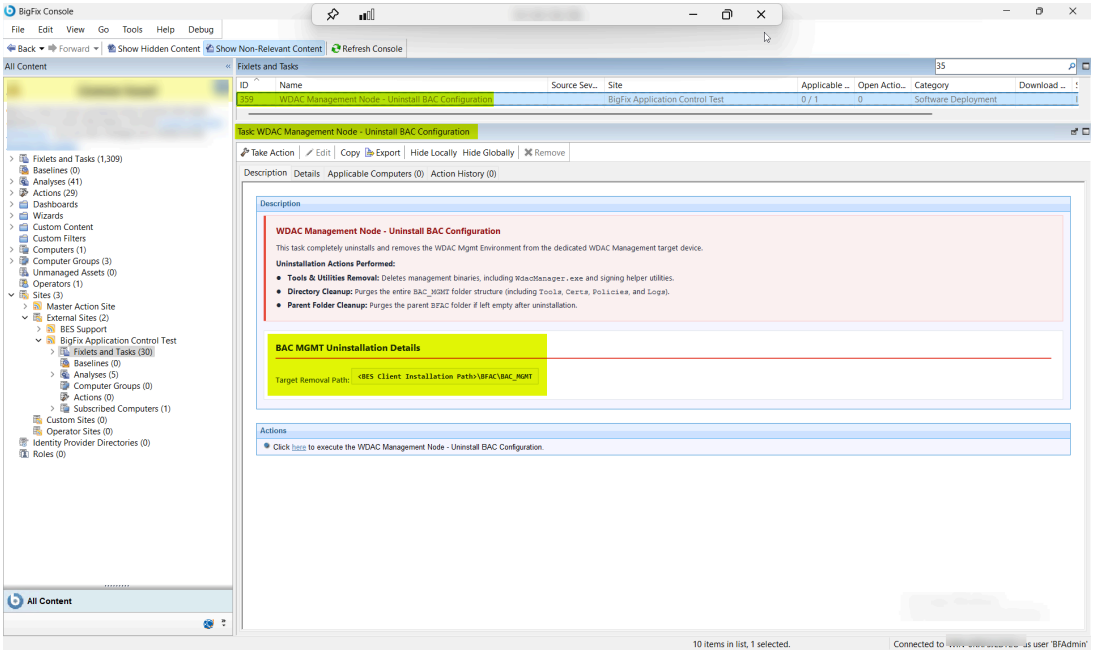
Remove the WDAC Management Environment from the dedicated WDAC Management Node by deleting management binaries including `WdacManager.exe` and signing helper utilities, purging the `BAC_MGMT` folder structure (Tools, Certs, Policies, and Logs — including the TPM-bound signing certificate stored in Certs), and removing the parent `BFAC` folder if left empty after uninstallation. Use **Target Removal Path** to specify the BAC_MGMT installation path, and select the **Cleanup Option** to choose between complete removal (all files and logs) or partial removal (preserving the Logs folder for audit retention).

This task (*ID 359*) completely uninstalls and removes the WDAC Management Environment from the dedicated WDAC Management target devices.

The task performs the following uninstall activities:

- **Tools & Utilities Removal**: Deletes management binaries, including `WdacManager.exe` and signing helper utilities.
- **Directory Cleanup**: Purges the `BAC_MGMT` folder structure (**Tools**, **Certs**, **Policies**, and **Logs**).
- **Parent Folder Cleanup**: Purges the parent `BFAC` folder if left empty after uninstallation.

Figure 24. BigFix Console — WDAC Management Node Uninstall BAC Configuration Task (ID 359) Overview



1. In the BigFix Console, navigate to **All Content > BigFix Application Control > Fixlets and Tasks**.
2. From the **Fixlets and Tasks** pane, select **Task: WDAC Management Node - Uninstall BAC Configuration**.
3. From the **Task: WDAC Management Node - Uninstall BAC Configuration** pane, under **BAC MGMT Uninstallation Details**, enter the following.

Figure 25. WDAC Management Node BAC Uninstall — Configuration Options: Target Removal Path and Cleanup Mode

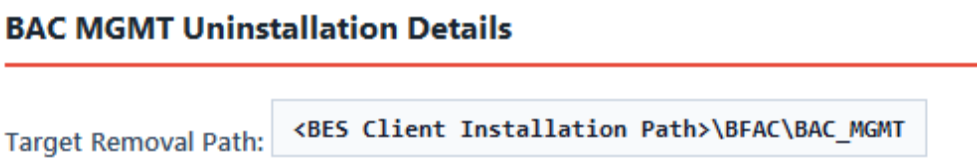


Table 20. WDAC Management Node BAC Uninstall — Configuration Options Field Descriptions

Field Name	Description
Target Removal Path	The full installation path of the `BAC_MGMT` folder on the WDAC Management Node — the directory specified when Task ID 352 (WDAC Management Node configuration) was installed. The task deletes the `BAC_MGMT` folder structure (Tools, Certs, Policies, and Logs subfolders), all binaries including `WdacManager.exe`, and the TPM-bound signing certificate stored in the Certs folder. The parent `BFAC` folder is also removed if left empty. Defaults to the standard BigFix Client installation path if left blank.

4. From the **Task: WDAC Management Node - Uninstall BAC Configuration** pane, click the **Applicable Computers(n)** tab and view the endpoints on which you want to run the task.
5. Select the **Take Actions** tab and select the endpoints on which you want to apply this task.
6. Click **OK**.

A successful execution of this task produces the following outcomes on the WDAC Management Node:

- **WdacManager.exe and signing utilities removed**: The `WdacManager.exe` binary and all associated signing helper utilities are deleted from the `BAC_MGMT\Tools` folder. Policy compilation and signing operations are no longer available on this device.
- **BAC_MGMT folder structure purged**: The complete `BAC_MGMT` directory — including the Tools, Certs, Policies, and Logs subfolders — is removed from the path specified in **Target Removal Path**. The TPM-bound signing certificate stored in the Certs folder is permanently deleted. If reinstalling, Task ID 353 must be re-run to re-provision the signing certificate after Task ID 352 completes.
- **BFAC parent folder conditionally removed**: If the parent `BFAC` folder is empty after the `BAC_MGMT` folder is removed, the `BFAC` folder is also deleted. If other contents remain in the `BFAC` folder (for example, the `BAC_SERVER` folder from the Root Server component), the parent folder is left in place.
- **WDAC Management Environment fully decommissioned**: The policy authoring, signing, and publishing infrastructure is no longer present on the Management Node. To reinstate it, re-run Task ID 352 (**Configure and Install the WDAC Management Node**) followed by Task ID 353 (**Provision the TPM-Enforced WDAC Policy Signing Certificate on the WDAC Management Node**).

Warning: The TPM-bound signing certificate in `BAC_MGMT\Certs` is permanently deleted by this task. Any WDAC policies already deployed to target endpoints will continue to be enforced using the previously distributed policy binaries, but no new or updated policies can be compiled and signed until the Management Node is reinstalled and the certificate is re-provisioned.

Installation Counterpart

Configure and Install the WDAC Management Node *(on page 41)*

Provision the TPM-Enforced WDAC Policy Signing Certificate on the WDAC Management Node *(on page 44)*

Related Uninstall Tasks

Uninstall WDAC Components from Target Endpoints *(on page 75)*

Uninstall BigFix Application Control Configuration from the Root Server *(on page 70)*

Deployment Context

Administering BigFix Application Control v2.x *(on page 35)*

System Architecture

Reference and Troubleshooting

BitLocker Recovery Key Requirements Before WDAC Recovery *(on page 25)*

BigFix Application Control v2.x: Frequently Asked Questions *(on page 141)*

BigFix Application Control: Troubleshooting WDAC Enforcement Issues *(on page 156)*

Uninstall WDAC Components from Target Endpoints

Remove all deployed signed WDAC policies from target endpoints following Microsoft legacy uninstall standards by fetching a signed `AllowAll_*.cip.p7` policy from the BigFix Custom Site identified by **Site Prefix**, deploying it using the filename specified in **AllowAll Policy Name** to overwrite active restrictive policies in System32 and EFI partitions, triggering `RefreshPolicy.exe` to apply the permissive overwrite immediately, purging all binaries and tools from the `\BFACCLIENT\BAC_CLIENT` directory, and enforcing a mandatory system reboot to flush kernel-cached policy states.

This task (*ID 360*) safely uninstalls deployed signed WDAC policies according to Microsoft legacy standards by fetching a signed AllowAll policy (.cip.p7), overwriting active policies in System32 and EFI, triggering live policy refresh via RefreshPolicy.exe, and clearing BAC Client directories before rebooting.

The task performs the following uninstall activities:

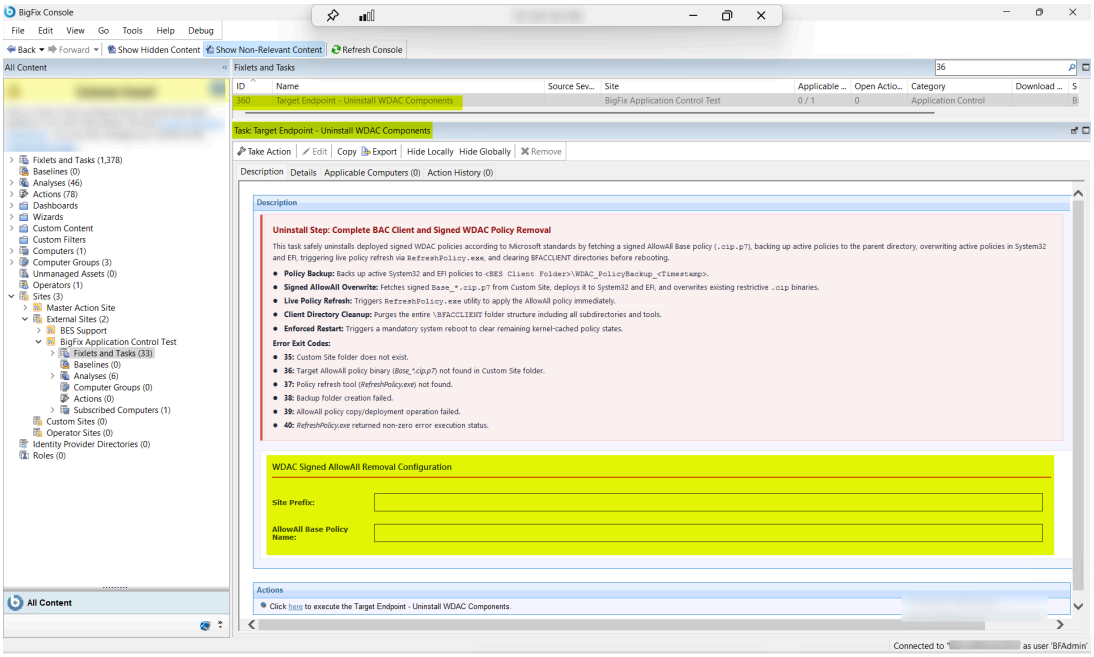
- **Policy Backup**: Backs up active System32 and EFI policies to `<BES Client Folder>\WDAC_PolicyBackup_<Timestamp>`.
- **Signed AllowAll Overwrite**: Fetches signed `AllowAll_*.cip.p7` from Custom Site, deploys it to System32 and EFI, and overwrites existing restrictive `.cip` binaries.
- **Live Policy Refresh**: Triggers `RefreshPolicy.exe` utility to apply the **AllowAll** policy immediately.
- **Client Directory Cleanup**: Purges binaries, tools, and folders from `\BFACCLIENT\BAC_CLIENT`.
- **Enforced Restart**: Triggers a mandatory system reboot to clear remaining kernel-cached policy states.

Refer to the table below to know more about the task's exit code.

Table 21. Task Exit Code Reference: Uninstall WDAC Components from Target Endpoints Codes (ID 360)

Exit Code	Description
35	Custom Site folder does not exist.
36	Target **AllowAll** policy binary (`Base_*.cip.p7`) not found in Custom Site folder.
37	Policy refresh tool (`RefreshPolicy.exe`) not found.
38	Backup folder creation failed.
39	**AllowAll** policy copy/deployment operation failed.
40	`RefreshPolicy.exe` returned non-zero error execution status.

Figure 26. BigFix Console — Target Endpoint Uninstall WDAC Components Task (ID 360) Overview



1. In the BigFix® Console, navigate to **All Content > BigFix Application Control > Fixlets and Tasks**.
2. From the **Fixlets and Tasks** pane, select **Task: Target Endpoint - Uninstall WDAC Components**.
3. From the **Task: Target Endpoint - Uninstall WDAC Components** pane, under **WDAC Signed AllowAll Removal Configuration**, enter the following.

Figure 27. Target Endpoint WDAC Uninstall — Configuration Options: Site Prefix and AllowAll Policy Name

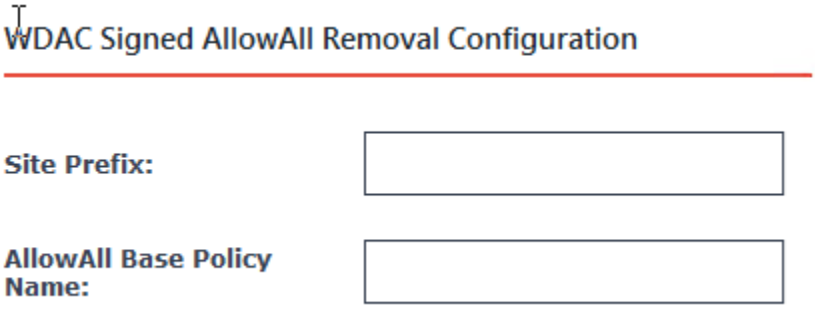


Table 22. Target Endpoint WDAC Uninstall — Configuration Options Field Descriptions

Field Name	Description
Site Prefix	The name prefix of the BigFix Custom Site hosting the signed `AllowAll_*.cip.p7` policy file — the Custom Site from which the task fetches the AllowAll overwrite policy. This must match the Custom Site created during Task ID 353 (TPM signing certificate provisioning) and used to publish WDAC policies. The task uses this prefix to construct the download URL for the AllowAll policy file.
AllowAll Policy Name	The exact filename of the signed AllowAll policy file (in `AllowAll_*.cip.p7` format) hosted on the Custom Site. This file was generated by the WDAC Management Node signing pipeline and published to the Custom Site for use in endpoint enforcement and uninstall procedures. The task fetches this file, deploys it to the `\Windows\System32\CodeIntegrity\CiPolicies\Active` directory and the EFI partition, and triggers `RefreshPolicy.exe` to apply it immediately.

4. From the **Task: Target Endpoint - Uninstall WDAC Components** pane, click the **Applicable Computers(n)** tab and view the endpoints on which you want to run the task.
5. Select the **Take Actions** tab and select the endpoints on which you want to apply this task.
6. Click **OK**.

A successful execution of this task produces the following outcomes on each target endpoint:

- **AllowAll policy deployed and active**: The signed `AllowAll_*.cip.p7` policy is fetched from the BigFix Custom Site, deployed to `\Windows\System32\CodeIntegrity\CiPolicies\Active` and the EFI partition, and the existing restrictive `.cip` policy binaries are overwritten. Application execution restrictions are immediately removed.
- **Policy refresh applied immediately**: `RefreshPolicy.exe` is triggered and the AllowAll policy is applied without waiting for the next Group Policy refresh cycle. All previously blocked applications are permitted to run on the endpoint.
- **BAC_CLIENT directory purged**: All BigFix Application Control client binaries, tools, and utilities are deleted from the `\BFACCLIENT\BAC_CLIENT` directory hierarchy. The BAC_CLIENT folder structure is removed from the endpoint.
- **Mandatory system reboot triggered**: The endpoint is restarted to flush any remaining kernel-cached WDAC policy states. After the reboot, no WDAC enforcement is active on the endpoint and all application execution restrictions are cleared.
- **Endpoint fully decommissioned from WDAC enforcement**: After the reboot, the target endpoint operates without any active WDAC policy. The endpoint is removed from the pool of WDAC-managed devices. To re-enroll, re-run Task ID 356 (**Set Up and Initialize WDAC on Target Endpoints**) followed by Task ID 357 (**Deploy and Refresh WDAC Policies on Target Endpoints**).

Important: This task overwrites EFI partition policy files as part of the AllowAll deployment. On BitLocker-encrypted endpoints, this may change the TPM boot measurement and trigger a BitLocker recovery key prompt on next boot. Have the 48-digit recovery key available for each target endpoint before running this task.

Installation Counterparts

Set Up and Initialize WDAC on Target Endpoints *(on page 64)*

Deploy and Refresh WDAC Policies on Target Endpoints *(on page 66)*

Remove WDAC Base and Supplemental Policies *(on page 126)*

Related Uninstall Tasks

Uninstall BigFix Application Control Configuration from the WDAC Management Node *(on page 73)*

Uninstall BigFix Application Control Configuration from the Root Server *(on page 70)*

Deployment Context

Administering BigFix Application Control v2.x *(on page 35)*

Reference and Troubleshooting

BitLocker Recovery Key Requirements Before WDAC Recovery *(on page 25)*

BigFix Application Control v2.x: Policy Behavior and Deployment Considerations *(on page 32)*

BigFix Application Control v2.x: Frequently Asked Questions *(on page 141)*

BigFix Application Control: Troubleshooting WDAC Enforcement Issues *(on page 156)*

Submit a WDAC XML Policy for Review

Submit an existing WDAC XML policy file from the WDAC Management Node to BigFix® for policy review by archiving and uploading the specified XML file from the `<BES Client Installation Path>\BFAC\BAC_MGMT\Policies\Working` directory. Base policy XML files must reside directly in the Working directory; supplemental policy XML files must reside in their parent base policy subdirectory under Working. Enter the **Policy File Name** with or without the `.xml` extension to identify which policy to submit.

This task (*ID 361*) submits an existing WDAC XML policy to BigFix for review by archiving and uploading the specified XML file directly from the endpoint.

The task has the following requirements and instructions for the review activity:

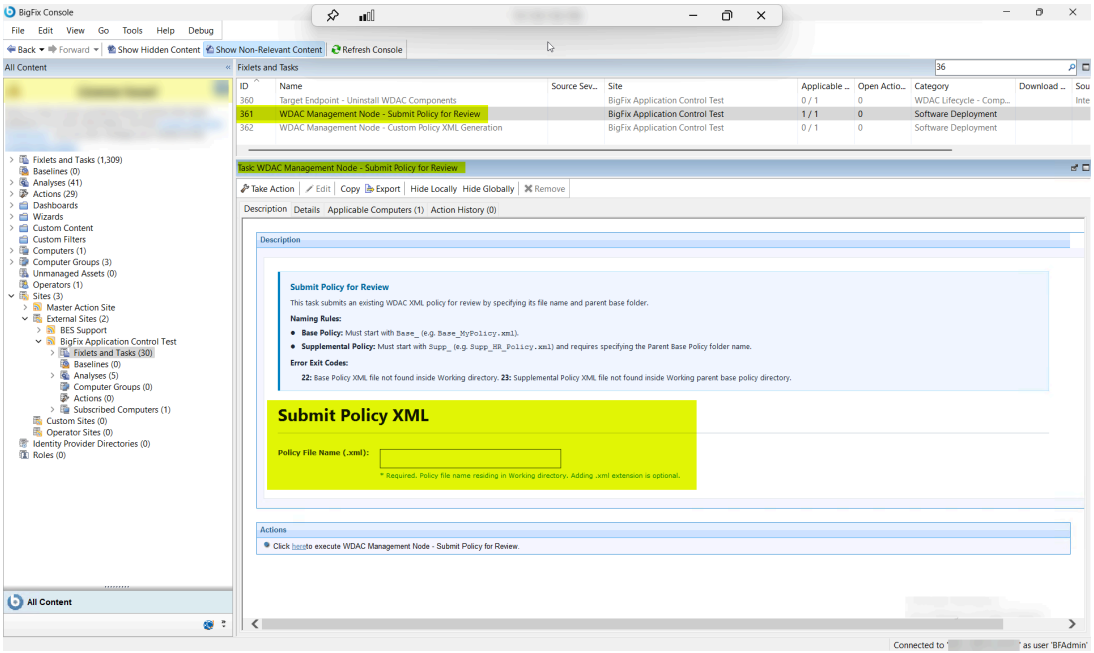
- **Policy Location**: The target XML file must reside in `<BES Client Installation Path>\BFAC\BAC_MGMT\Policies\Working`.
- **Policy Name Input**: Enter only the file name (e.g. `MyPolicy` or `MyPolicy.xml`). Specifying the `.xml` extension is optional.

Refer to the table below to know more about the task's exit code.

Table 23. Task Exit Code Reference: Submit Policy Failure Codes

Exit Code	Description
22	Base Policy XML file not found inside Working directory.
23	Supplemental Policy XML file not found inside Working parent base policy directory.

Figure 28. BigFix Console — Submit Policy for Review Task (ID 361) Overview



1. In the BigFix Console, navigate to **All Content > BigFix Application Control > Fixlets and Tasks**.
2. From the **Fixlets and Tasks** pane, select **Task: Submit Policy for Review**.
3. From the **Task: Submit Policy for Review** pane, under **Submit Policy XML**, enter the following.

Figure 29. Submit Policy for Review — Submit Policy XML: Policy File Name Configuration

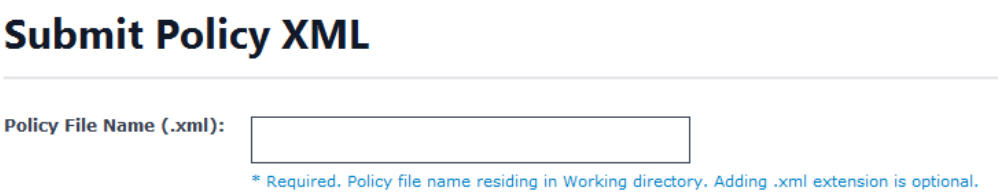


Table 24. Submit Policy for Review — Configuration Options Field Descriptions

Field Name	Description
Policy File Name (.xml)	The file name of the WDAC XML policy to archive and upload for review. Base policy XML files must reside directly in `BFAC\BAC_-MGMT\Policies\Working`; supplemental policy XML files must reside in a subdirectory named after the parent base policy (for example, `BFAC\BAC_MGMT\Policies\Working\Base_CorpPolicy\Supp_Finance.xml`). Enter the file name only — do not include the directory path. The `.xml` extension is optional. If the policy file is not found in the expected location, the task exits with code 22 (base policy) or 23 (supplemental policy).

4. From the **Task: Submit Policy for Review** pane, click the **Applicable Computers(n)** tab and view the endpoints on which you want to run the task.
5. Select the **Take Actions** tab and select the endpoints on which you want to apply this task.
6. Click **OK**.

A successful execution of this task produces the following outcomes on the WDAC Management Node:

- **Policy XML file archived**: The specified WDAC XML policy file is located in the `BFAC\BAC_MGMT\Policies\Working` directory (or its parent base policy subdirectory for supplemental policies), archived, and prepared for upload.
- **Policy XML uploaded to BigFix**: The archived policy XML file is uploaded directly from the WDAC Management Node to BigFix®. The policy is now available for review within the BigFix® Application Control policy review workflow.
- **Policy available for review and refinement**: Reviewers can examine the submitted XML policy definition alongside audit-mode blocked-application event data collected by Task ID 357 on target endpoints. The review process informs whether the policy rules require refinement before transitioning to enforcement mode.

Tip: If the task exits with code 22 or 23, verify that the policy XML file exists in the correct Working directory location before re-running. Base policy files must be directly in `BAC_MGMT\Policies\Working`; supplemental policy files must be in a subdirectory named after their parent base policy.

Policy Creation Prerequisites

Create and Manage WDAC Policies on the WDAC Management Node *(on page 49)*

Compile, Sign, and Publish WDAC Policies from the WDAC Management Node *(on page 60)*

Deployment Context

Deploy and Refresh WDAC Policies on Target Endpoints *(on page 66)*

Administering BigFix Application Control v2.x *(on page 35)*

System Architecture

Post-Review Actions and Policy Lifecycle

Remove WDAC Base and Supplemental Policies *(on page 126)*

BigFix Application Control v2.x: Policy Behavior and Deployment Considerations *(on page 32)*

Reference and Troubleshooting

BitLocker Recovery Key Requirements Before WDAC Recovery *(on page 25)*

BigFix Application Control v2.x: Frequently Asked Questions *(on page 141)*

BigFix Application Control: Troubleshooting WDAC Enforcement Issues *(on page 156)*

Generate a Custom WDAC XML Policy on the WDAC Management Node

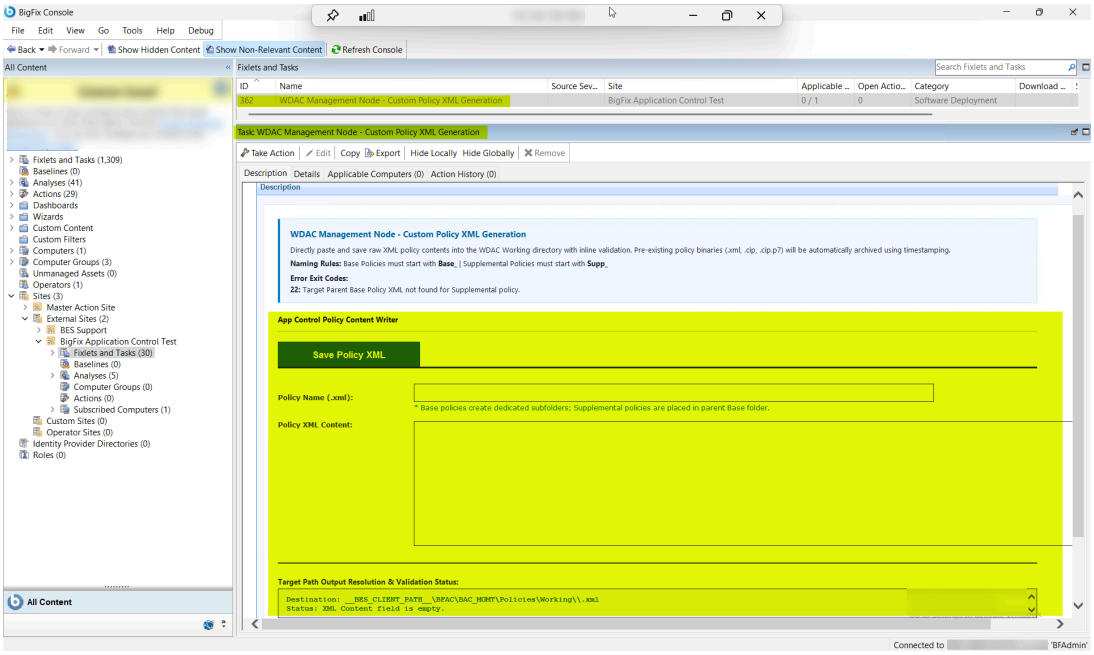
Generate a custom WDAC XML policy directly on the WDAC Management Node by generating a raw XML or by editing an existing policy XML using the App Control Policy Wizard content into the **Policy XML Content** field and saving it to the `<BES Client Installation Path>\BFAC\BAC_MGMT\Policies\Working` directory with inline validation. Pre-existing policy binaries (`.xml`, `.cip`, `.cip.p7`) at the same path are automatically archived with timestamps before the new file is saved. Policy names must follow the `Base_` prefix convention for base policies and the `Supp_` prefix convention for supplemental policies.

Note: To generate a raw XML or editing an existing policy XML, refer to App Control Policy Wizard *(on page 105)*.

Use this task to directly paste and save raw XML policy contents into the WDAC Working directory with inline validation. Pre-existing policy binaries (**.xml**, **.cip**, **.cip.p7**) will be automatically archived using time stamps. The task has the following naming requirements:

- **Naming Rules:** Base policies must start with **Base_** and the supplemental policies must start with **Supp_**.

Figure 30. BigFix Console — WDAC Management Node Custom Policy XML Generation Task (ID 362) Overview



Refer to the following exit code table for troubleshooting information if the task fails:

Table 25. Custom Policy XML Generation — Task Exit Code Reference

Exit Code	Description
22	Target Parent Base Policy XML not found for Supplemental policy.

1. In the BigFix® Console, navigate to **All Content > BigFix Application Control > Fixlets and Tasks**.
2. From the **Fixlets and Tasks** pane, select **Task: WDAC Management Node - Custom Policy XML Generation**.
3. From the **Task: WDAC Management Node - Custom Policy XML Generation** pane, under **Save Policy XML**, enter the following.

Figure 31. WDAC Management Node Custom Policy XML Generation — Configuration Options: Policy Name and Policy XML Content

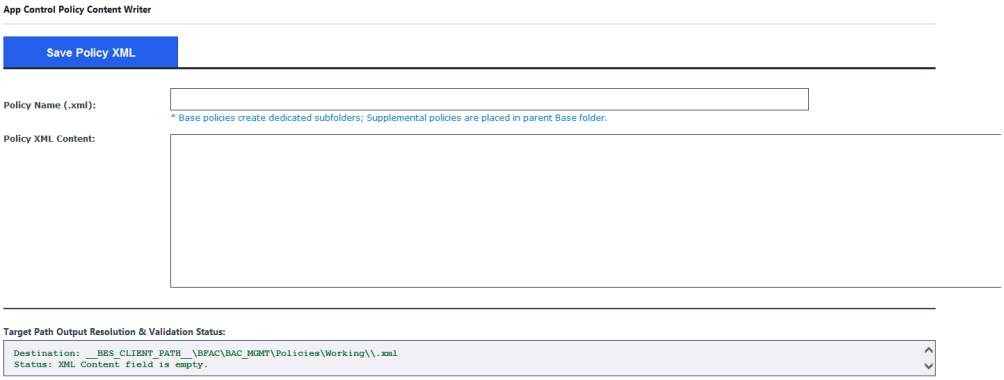


Table 26. Custom Policy XML Generation — Configuration Options Field Descriptions

Field Name	Description
Policy Name (.xml)	The file name for the new WDAC XML policy. Base policies must begin with `Base_` (for example, `Base_CorpPolicy`); supplemental policies must begin with `Supp_` followed by a name that identifies the supplemental scope (for example, `Supp_Finance`). The `.xml` extension is optional. If a policy file with the same name already exists in the Working directory, it is automatically archived with a timestamp before the new file is saved. If creating a supplemental policy, the parent base policy XML must already exist in the Working directory — otherwise the task exits with code 22.
Policy XML Content	The raw WDAC XML policy body to save to the Working directory. Paste the full XML content of the policy directly into this field. The task performs inline validation on the pasted XML before saving. If the file cannot be written to disk (for example, due to a permissions issue or disk error), the task exits with code 35.

4. From the **Task: WDAC Management Node - Custom Policy XML Generation** pane, click the **Applicable Computers(n)** tab and view the endpoints on which you want to run the task.

5. Select the **Take Actions** tab and select the endpoints on which you want to apply this installer task.
6. Click **OK**.

A successful execution of this task produces the following outcomes on the WDAC Management Node:

- **Custom policy XML file saved**: The pasted XML content passes inline validation and is saved as a new `<PolicyName>.xml` file in the `BAC_MGMT\Policies\Working` directory. Base policies are saved directly in Working; supplemental policies are saved in a subdirectory named after the parent base policy.
- **Pre-existing binaries archived**: If a policy file with the same name already existed in the Working directory, the previous `.xml`, `.cip`, and `.cip.p7` binaries are automatically archived with timestamps before the new file is written. No policy data is permanently overwritten without an archive copy.
- **Policy ready for compile-sign-publish pipeline**: The saved XML file is now available in the Working directory for Task ID 355 (Compile, Sign, and Publish) to process into a signed `.cip.p7` binary and publish to the BigFix Custom Site.
- **Policy available for review submission**: The saved XML file can be submitted for policy review via Task ID 361 (Submit a WDAC XML Policy for Review) directly from the Working directory.

Tip: If the task exits with code 22, verify that the parent base policy XML file exists in `BAC_MGMT\Policies\Working` before re-running for a supplemental policy. If the task exits with code 35, check disk space and write permissions on the Working directory path.

Policy Authoring Prerequisites

Create and Manage WDAC Policies on the WDAC Management Node *(on page 49)*

Configure and Install the WDAC Management Node *(on page 41)*

Next Steps in the Policy Lifecycle

Compile, Sign, and Publish WDAC Policies from the WDAC Management Node *(on page 60)*

Deploy and Refresh WDAC Policies on Target Endpoints *(on page 66)*

Submit a WDAC XML Policy for Review *(on page 79)*

Policy Removal and Replacement

Remove WDAC Base and Supplemental Policies *(on page 126)*

Reference and Troubleshooting

BigFix Application Control v2.x: Policy Behavior and Deployment Considerations *(on page 32)*

BigFix Application Control v2.x: Frequently Asked Questions *(on page 141)*

BigFix Application Control: Troubleshooting WDAC Enforcement Issues *(on page 156)*

Sync Published Policy XML to the WDAC Management Node

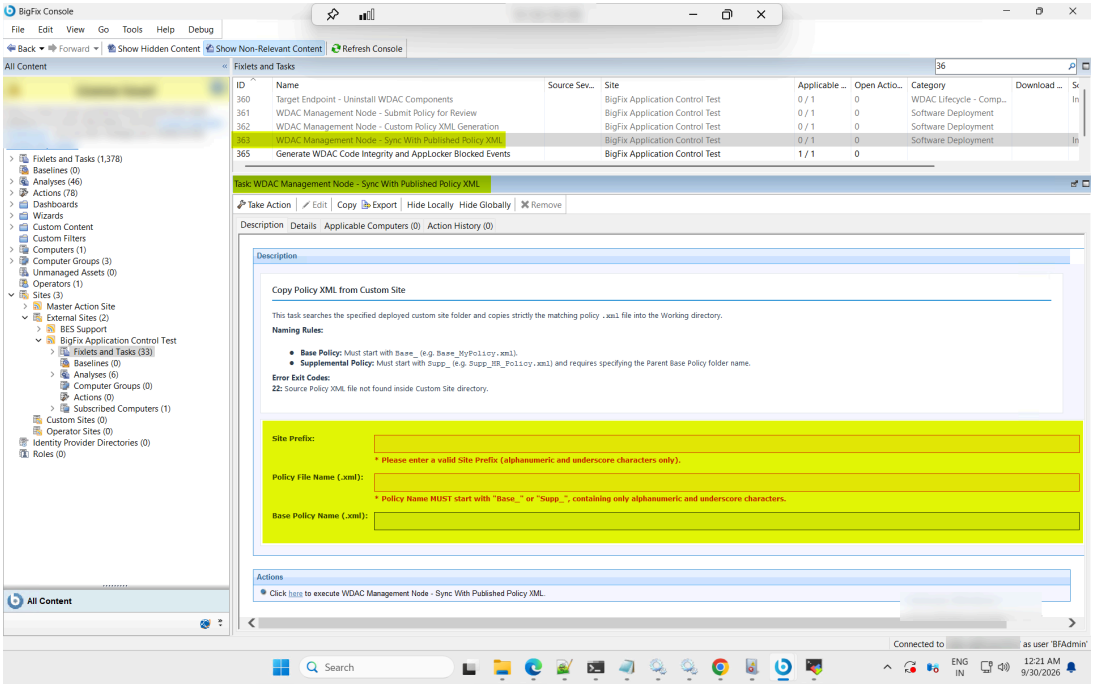
Copy a published policy XML file from a deployed custom site folder into the WDAC Management Node working directory, with automatic archival of any existing working folder contents, in preparation for compiling, signing, and publishing the policy.

This task (*ID 363*) searches the specified deployed custom site folder and copies strictly the matching policy `.xml` file into the working directory.

Naming Rules:

- **Base Policy**: Must start with *Base_* (e.g. `Base_MyPolicy.xml`).
- **Supplemental Policy**: Must start with *Supp_* (e.g. `Supp_HR_Policy.xml`) and requires specifying the parent base policy folder name.

Figure 32. Sync Published Policy XML to the WDAC Management Node — Task Overview



Refer to the following exit code table for troubleshooting information if the task fails:

Table 27. Task Exit Code Reference: WDAC Management Node — Sync With Published Policy XML (ID 363) Failure Codes

Exit Code	Description
22	Source Policy XML file not found inside Custom Site directory.

1. In the BigFix® Console, navigate to **All Content > BigFix Application Control > Fixlets and Tasks**.
2. From the **Fixlets and Tasks** pane, select **Task: WDAC Management Node - Sync With Published Policy XML**.

3. From the **Task: WDAC Management Node - Sync With Published Policy XML** pane, under **Copy Policy XML from Custom Site**, enter the following.

Figure 33. Sync Published Policy XML to the WDAC Management Node (ID 363) — Copy Policy XML from Custom Site Options

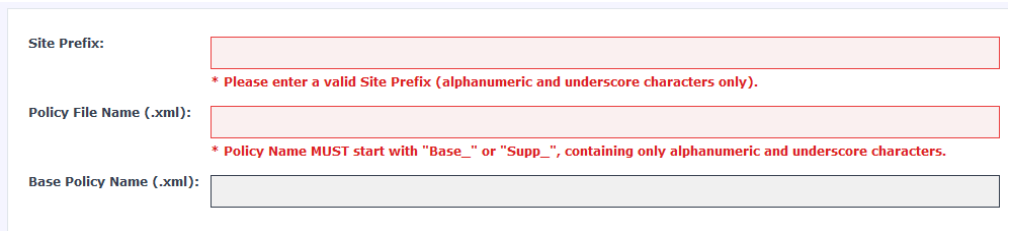


Table 28. Sync Published Policy XML — Copy Policy XML from Custom Site: Field Descriptions

Field Name	Description
Site Prefix	The prefix of the deployed custom site from which the policy XML file will be copied. Used to locate the correct custom site folder on the WDAC Management Node.
Policy File Name (.xml)	The file name of the policy XML to copy into the working directory, including the `.xml` extension. Must follow the naming rules: Base policies must start with `Base_` and Supplemental policies must start with `Supp_`.
Base Policy Name (.xml)	Required for Supplemental policies only. The file name of the parent Base Policy, used to locate the correct source subdirectory within the custom site folder.

4. From the **Task: WDAC Management Node — Sync With Published Policy XML** pane, click the **Applicable Computers(n)** tab and view the endpoints on which you want to run the task.
5. Select the **Take Actions** tab and select the endpoints on which you want to apply this installer task.
6. Click **OK**.

The specified policy XML file is copied from the deployed custom site folder into the WDAC Management Node working directory. Any pre-existing working folder contents are automatically archived before the copy completes. The policy XML is now available in the working directory for the next step in the policy lifecycle.

- To compile, sign, and publish the synced policy, run the Compile, Sign, and Publish WDAC Policies task (Task ID 355) on the WDAC Management Node.
- If the task fails, refer to the exit code table in the task context for troubleshooting: exit code 22 indicates the source XML was not found in the specified custom site folder; exit code 24 indicates the working folder archive backup failed.

Policy XML Authoring

Create and Manage WDAC Policies on the WDAC Management Node *(on page 49)*

Generate a Custom WDAC XML Policy on the WDAC Management Node *(on page 82)*

Policy Compilation and Deployment

Compile, Sign, and Publish WDAC Policies from the WDAC Management Node *(on page 60)*

Deploy and Refresh WDAC Policies on Target Endpoints *(on page 66)*

Review and Reference

Submit a WDAC XML Policy for Review *(on page 79)*

BigFix Application Control v2.x: Policy Behavior and Deployment Considerations *(on page 32)*

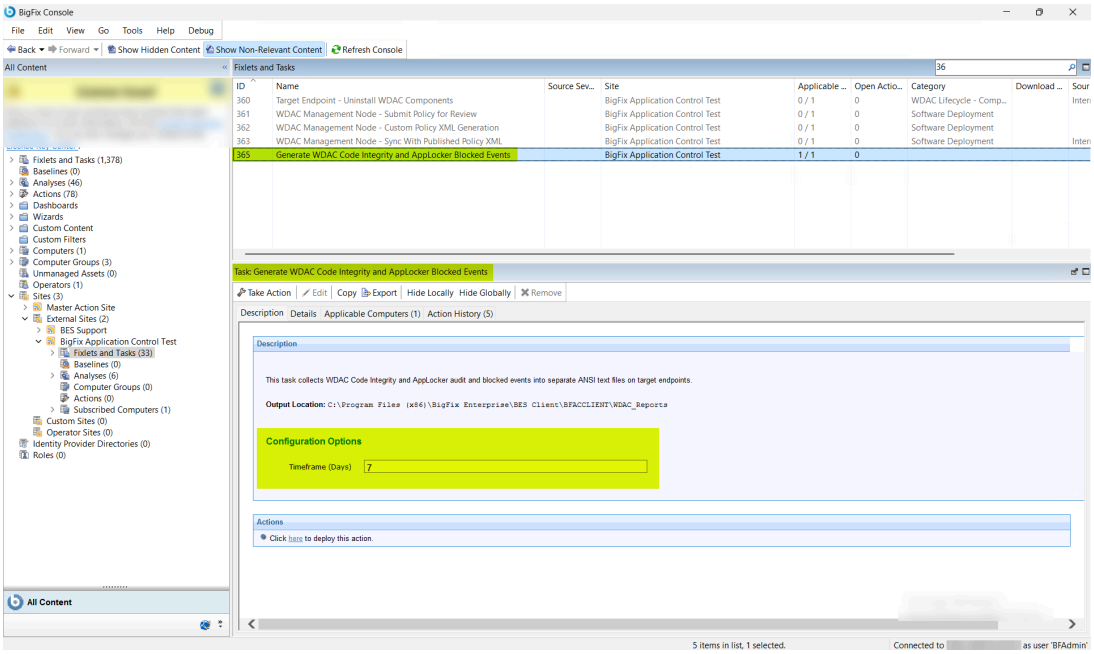
Generate WDAC Code Integrity and AppLocker Blocked Events

Collect WDAC Code Integrity and AppLocker audit and blocked events from target endpoints into ANSI text files, stored in the WDAC_Reports folder on each endpoint, for review through BigFix® Web Reports or the Application Control reporting mechanism.

This task (*ID 365*) collects WDAC Code Integrity and AppLocker audit and blocked events into separate ANSI text files on target endpoints.

Output Location: `C:\Program Files (x86)\BigFix Enterprise\BES Client\BFACCLIENT\WDAC_Reports`

Figure 34. Generate WDAC Code Integrity and AppLocker Blocked Events — Task Overview (ID 365)



1. In the BigFix® Console, navigate to **All Content > BigFix Application Control > Fixlets and Tasks**.
2. From the **Fixlets and Tasks** pane, select **Task: Generate WDAC Code Integrity and AppLocker Blocked Events**.
3. From the **Task: Generate WDAC Code Integrity and AppLocker Blocked Events** pane, under **Configuration Options**, enter the following.

Figure 35. Generate WDAC Code Integrity and AppLocker Blocked Events (ID 365) — Configuration Options



Table 29. Generate WDAC Code Integrity and AppLocker Blocked Events — Configuration Options Field Descriptions

Field Name	Description
Timeframe (Days)	Number of days of event history to collect from each target end-point. The task queries the endpoint's event log within this window and writes matching WDAC Code Integrity and AppLocker events to the output files in the WDAC_Reports folder.

4. From the **Task: WDAC Management Node - Custom Policy XML Generation** pane, click the **Applicable Computers(n)** tab and view the endpoints on which you want to run the task.
5. Select the **Take Actions** tab and select the endpoints on which you want to apply this installer task.
6. Click **OK**.

WDAC Code Integrity and AppLocker audit and blocked events are collected from each target endpoint and written to separate ANSI text files in the `WDAC_Reports` folder. The report files are available for review through BigFix® Web Reports or the applicable Application Control reporting mechanism.

- Use the collected event data to identify application files or components that are audited or blocked, and to determine the appropriate WDAC rule type for those components.
- If the collected events indicate that a policy adjustment is required, update the WDAC policy on the WDAC Management Node and redeploy to the affected endpoints.

Event Review and Reporting

Submit a WDAC XML Policy for Review *(on page 79)*

View Endpoint Details from BigFix Web Reports *(on page 131)*

Policy Analysis and Rule Adjustment

Best Practices for WDAC Policy Rule Validation *(on page 11)*

Create and Manage WDAC Policies on the WDAC Management Node *(on page 49)*

Conceptual Reference

BigFix Application Control v2.x: Policy Behavior and Deployment Considerations *(on page 32)*

WDAC Rule Types and Use Cases *(on page 8)*

Chapter 11. Deprecated BigFix Application Control v2.x Fixlets and Tasks

This topic lists ten deprecated BigFix Application Control v2.x Fixlets and Tasks that are scheduled for removal in a future release. The deprecated tasks cover self-signed certificate setup and renewal, Microsoft default and custom WDAC base policy deployment, supplemental policy deployment, certificate thumbprint refresh, BAC folder ACL enforcement, blocked WDAC event log generation, base and supplemental policy removal, and full WDAC component removal. Use the current administration tasks in Administering Application Control v2.x *(on page 35)* instead.

All the tasks described in this section are to be performed by users with administrator role for installing, configuring, and removing BigFix Application Control v2.x.

Important: All the tasks listed in this section are deprecated and will be removed from BigFix Application Control v2.x in near future.

The following tasks are covered under this topic:

- **Create & Setup Self-Signed Certificate on Endpoint v2.x**:

  This task generates and configures a self-signed code-signing certificate on the endpoint for use with Windows Defender Application Control (WDAC).

- **Deploy Default Microsoft Base Policy v2.x**:

  This task deploys a signed Windows Defender Application Control (WDAC) Base Policy on an endpoint using a secure and controlled workflow. The policy is selected from a list of Microsoft-recommended baseline policies and applied as the system's active base policy.

- **Deploy Custom Base Policy v2.x**:

  This task deploys a Custom Windows Defender Application Control (WDAC) Base Policy on the endpoint. The raw XML configuration for the policy must be provided before taking action.

- **Deploy Supplemental Policy on Endpoint v2.x**:

  This task creates and deploys a signed Windows Defender Application Control (WDAC) supplemental policy on the endpoint using a controlled and secure approach. The supplemental policy is generated as a separate policy file and linked to an existing base policy using the provided Base Policy GUID.

- **Refresh Self-Signed Certificate and Update Deployed Policies Signers on Endpoint v2.x**:

  This task automates the end-to-end renewal, signing, and deployment of Windows Defender Application Control (WDAC) policies to ensure policies remain active and securely signed.

- **Refresh Thumbprint of Deployed Policies Signers on Endpoint v2.x**:

  This task automates the taking of the thumbprint of the new certificates and then signs them to ensure policies remain active and securely signed.

- **Enforce Secure ACL on BAC Folder on Endpoint v2.x**:

This task secures the BigFix Application Control (BAC) directory by enforcing strict access control and ownership settings.

- **Generate Blocked WDAC Event Logs on Endpoint v2.x**:

  This task extracts Windows Defender Application Control (WDAC) / App Control block events from endpoint event logs for both audit and enforced modes, and generates a structured JSON report for analysis.

- **Remove (Base/Supplemental) Policy from Endpoint v2.x**:

  This task manages Windows Defender Application Control (WDAC) policies on the endpoint by supporting both supplemental policy removal and a full base policy reset using a controlled and safe approach.

- **Remove WDAC Components from Endpoint v2.x**:

  This task removes all the associated files and folders related to Windows Defender Application Control (WDAC).

Current Replacement Administration Tasks

Administering BigFix Application Control v2.x *(on page 35)*

Provision the TPM-Enforced WDAC Policy Signing Certificate on the WDAC Management Node *(on page 44)*

Create and Manage WDAC Policies on the WDAC Management Node *(on page 49)*

Deprecated Certificate and Policy Deployment Tasks

Create and Set Up a Self-Signed Certificate on Endpoint v2.x *(on page 98)*

Deploy Default Microsoft Base Policy v2.x *(on page 101)*

Deploy Supplemental Policy on Endpoint v2.x *(on page 113)*

Deprecated Monitoring and Removal Tasks

Generate Blocked WDAC Event Logs on Endpoint v2.x *(on page 123)*

Remove Base and Supplemental Policy from Endpoint v2.x *(on page 126)*

Remove WDAC Components from Endpoint v2.x *(on page 129)*

Reference

BigFix Application Control v2.x Tasks: Exit Codes *(on page 134)*

BigFix Application Control v2.x: Frequently Asked Questions *(on page 141)*

BigFix Application Control: Troubleshooting WDAC Enforcement Issues *(on page 156)*

BigFix Application Control v2.0.0: WDAC Use Cases and Enforcement Model

BigFix Application Control v2.0.0 leverages Windows Defender Application Control (WDAC) to enforce kernel-level application allowlisting across managed Windows endpoints. This topic covers the WDAC enforcement model, the BigFix–WDAC integration architecture, foundational infrastructure provisioning using endpoint-based self-signed certificates, and the end-to-end Base Policy and Supplemental Policy lifecycle.

WDAC Enforcement Model

Windows Defender Application Control (WDAC) operates at the kernel level through the Code Integrity engine. A WDAC policy is authored in XML, compiled to a binary (`.cip`) file, and digitally signed. The signed policy is deployed to:

- `%OSDrive%\Windows\System32\CodeIntegrity\CiPolicies\Active\{PolicyId}.cip`
- `<EFI System Partition>\Microsoft\Boot\CiPolicies\Active\{PolicyId}.cip`

The OS kernel validates the policy at boot and at runtime. Only code that is explicitly allowed by the active policy is permitted to execute.

If the Base Policy is tampered with or becomes invalid (for example, deleted, modified, replaced, or signed by an untrusted certificate), the Code Integrity engine may take one or more of the following actions:

- Block policy loading.
- Prevent execution of unsigned code.
- Enter a recovery state.

In the worst case, a tampered or invalid Base Policy results in a system boot failure (Blue Screen of Death, BSOD), which may require secure physical intervention to recover.

Kernel-Level Policy Verification

The kernel verifies the following conditions before applying a WDAC policy:

- **Digital Signature Validity:** The policy must be correctly signed.
- **Trusted Signer:** The signing certificate must chain to a trusted root.
- **Version Monotonicity:** The incoming Base Policy version must be greater than or equal to the currently active version (prevents rollback attacks).
- **Policy Structure and GUID Integrity:** A Supplemental Policy must correctly reference the Base Policy GUID.
- **Enforcement State Consistency:** The enforcement mode must be valid and intact.

Important:

- Any kernel verification failure results in the policy being rejected and execution being blocked.
- A BSOD occurs only when the Base Policy fails verification. Supplemental Policy failures do not cause boot issues.

BigFix–WDAC Integration Overview

BigFix orchestrates the full WDAC policy lifecycle for application control across managed endpoints. The integration covers policy generation, deployment, health monitoring, and on-demand telemetry collection.

Policy Generation and Deployment

Base and Supplemental WDAC policies are generated locally on each endpoint using TPM-backed self-signed keys. The Custom Site hosts the approved application inventory file (`apps.csv`), while the External Site hosts the Default Policy XML template used as the starting point for Base Policy generation.

BigFix Analysis

Three types of BigFix Analysis support the integration:

- **Health Analysis**

  Detects missing or tampered WDAC policies and classifies each endpoint into one of three states: *Healthy*, *Drift*, or *Recovery Eligible*.

- **Reporting Analysis**

  Collects active policy GUIDs and version numbers, along with Code Integrity event logs. This analysis is globally activated to provide immediate forensic visibility following on-demand telemetry collection.

- **On-Demand Telemetry**

  Blocked execution event data is collected via BigFix Tasks and stored in ACL-hardened folders. The collection process automatically self-heals to ensure performance, data integrity, and a centralized audit trail.

Known Limitations and Risks

- Local administrators can remove self-signed certificates from an endpoint. This disrupts policy update orchestration; however, the existing EFI-resident policy remains enforced until the next successful policy update.
- EFI path misconfiguration can result in a BSOD across affected endpoints at scale.
- Covers policy deployment and enforcement using self-signed certificates only. Emergency policy procedures and recovery scenarios are out of scope for the MVP.

Use Case 1: Foundational Infrastructure Provisioning

Objective: Enable trusted, decentralized application control using endpoint-based certificates.

In this use case, each managed endpoint generates and uses its own self-signed certificate for local WDAC policy validation. Policy governance is handled centrally through Custom Site orchestration, while policy artifacts are hosted centrally and enforced locally on each endpoint.

Governance Site Components

The Custom Site provides the following artifacts used for policy governance:

- `apps.csv`

  The approved application inventory. This file defines which applications are permitted to execute across managed endpoints.

- `{custom}_policy.xml`

  The application control enforcement rules. This XML file defines the allowlist and blocklist rules that form the basis of the generated WDAC policy.

Outcomes

- Decentralized trust model — each endpoint maintains its own signing certificate.
- Simplified infrastructure provisioning — no centralized certificate authority is required.
- Scalable endpoint-level enforcement — policies are generated and enforced independently on each managed endpoint.

The infrastructure setup workflow is initiated by the Master Operator, who uploads `apps.csv` to the Custom Site. This single action makes the approved application inventory available to all subscribed endpoints for subsequent policy generation tasks.

Use Case 2: Policy Enforcement – Base and Supplemental Policies

Objective: Establish a hardware-bound, integrity-enforced WDAC foundation operating in a fail-closed model, enabling controlled and governed application management.

Base Policy Execution

The Base Policy task executes the following steps on each managed endpoint:

1. Purges unauthorized `.cip` files from `<EFI System Partition>\EFI\Microsoft\Boot\CiPolicies\Active\` to establish a clean and consistent baseline before deployment.
2. Retrieves the WDAC Default Template XML from the External Site, scans currently installed applications, and merges the results using PowerShell (New-CIPolicy and Merge-CIPolicy) to build the Base Policy.
3. Generates a unique self-signed signing key — TPM-backed if a TPM is available, otherwise software-stored. Signs the compiled policy, deploys the `.cip` file to the EFI partition, and triggers RefreshPolicy to activate enforcement.

Supplemental Policy Updates

Supplemental Policy updates follow this process on each endpoint:

1. Retrieve `apps.csv` from the Custom Site.
2. Generate the Supplemental Policy XML, referencing the active Base Policy GUID.
3. Apply the update only if the incoming version number is greater than the currently active version (anti-rollback enforcement).
4. Sign the Supplemental Policy using the TPM-backed key if available, otherwise using the software-stored key.
5. Apply the signed policy and trigger RefreshPolicy. The WDAC kernel enforces the updated rules immediately.

Security Highlights

- **Hardware-pinned enforcement:** TPM-backed key generation ensures that each endpoint's signing key is unique and hardware-bound, preventing key reuse or transfer across machines.
- **Anti-rollback:** Monotonic version enforcement prevents older, potentially less secure policy versions from being reinstated.
- **In-memory policy management:** Policies are managed in memory during generation and signing; no plaintext policy content is written to disk during the process.

Base Policy Generation Workflow

The following steps describe the end-to-end Base Policy generation workflow initiated from the BigFix Console:

1. The Master Operator triggers the Base Policy Task from the BigFix Console, either by selecting a pre-defined Base Policy or by providing a custom policy XML.
2. The BigFix Root Server dispatches the task/fixlet to the target Endpoint Device.
3. The Endpoint Device executes the Base Policy task:
    a. Initiates Base Policy generation using the custom XML or the default WDAC Policy XML.
    b. Purges unauthorized policy files from the EFI partition.
    c. Validates the policy version (anti-rollback check).
    d. Signs the compiled policy.
    e. Applies the signed policy and calls RefreshPolicy.
4. WDAC enforces the Base Policy on the endpoint.
5. The Supplemental Policy is enforced in conjunction with the Base Policy.

Supplemental Policy Generation Workflow

The following steps describe the end-to-end Supplemental Policy generation workflow:

1. The Master Operator triggers the Supplemental Policy Task from the BigFix Console, either by passing a parameter directly or by pointing to `apps.csv`.
2. The BigFix Root Server dispatches the task/fixlet to the target Endpoint Device.
3. The Endpoint Device executes the Supplemental Policy task:
    a. Initiates Supplemental Policy generation using the current Base Policy and the contents of `apps.csv`.
    b. Validates the incoming policy version against the active version (anti-rollback check).
    c. Signs the compiled Supplemental Policy.
    d. Applies the signed policy and calls RefreshPolicy.
4. WDAC enforces the updated Supplemental Policy rules on the endpoint.
5. The Supplemental Policy operates in conjunction with the Base Policy to determine the final set of allowed and blocked applications.

Product Context

BigFix Application Control: Product Overview

System Architecture

User Personas and Roles

WDAC Limitations *(on page 18)*

Behavior Considerations *(on page 32)*

Installation and Configuration

BigFix Application Control: Configuration and Installation *(on page 37)*

WDAC Management Node: Configuration and Installation *(on page 41)*

Provisioning the TPM-Backed Code-Signing Certificate *(on page 44)*

Setting Up and Initializing WDAC on Target Endpoints *(on page 64)*

Policy Deployment Tasks

Compiling, Signing, and Publishing WDAC Policies *(on page 60)*

Deploying and Refreshing WDAC Policies on Endpoints *(on page 66)*

Deploying the Default Microsoft Base Policy *(on page 101)*

Creating a New Base WDAC Policy *(on page 105)*

Deploying a Supplemental Policy *(on page 113)*

Troubleshooting and FAQs

Frequently Asked Questions *(on page 141)*

Troubleshooting Application Control *(on page 156)*

Migrating from BigFix Application Control v1.0.0 to v2.0.0

Migrating from Application Control v1.0.0 to v2.0.0 replaces the custom BAC rule engine with WDAC-based enforcement. Existing allow rules are recreated as Supplemental Policies, block rules are not migrated, and the process is a one-time, non-reversible operation per endpoint. This topic provides a six-step migration flow summary and key considerations before you begin.

As BigFix Application Control v2.0.0 is the only currently supported version of Application Control, it is strongly recommended that you migrate to Application Control v2.0.0. This migration replaces the Application Control v1.0.0 *custom BAC rule engine* with Application Control v2.0.0 *WDAC based enforcement*. Existing allow rules are recreated as **Supplemental Policies**. This migration is a one-time, non-reversible per endpoint without redeployment process.

Migration Flow Summary

Step 1: Export AC v1.0.0 Rules → **Step 2:** Remove AC v1.0.0 solution from endpoints → **Step 3:** Create & setup self-signed certificate → **Step 4:** Deploy AC v2.0.0 base policy → **Step 5:** Create supplemental policies → **Step 6:** Monitor via web reports

Remember:

- ◦ Migration is a **one-time** process.
  ◦ Base policy once set **cannot be switched**.
  ◦ Supplemental policies only extends the allowlist.
  ◦ WDAC uses a **default deny** enforcement model.
  ◦ Block rules are not migrated.
  ◦ Application Control supports only one base policy (MVP).

Refer to Migrate from BigFix Application Control v1.0.0 to v2.x topic for more details.

Product Context

BigFix Application Control: Product Overview

System Architecture

WDAC Rule Types and Use Cases *(on page 8)*

BigFix Application Control: WDAC Known Limitations *(on page 18)*

Migration Step Tasks

Creating and Setting Up the Self-Signed Certificate *(on page 98)*

Deploying the Default Microsoft Base Policy *(on page 101)*

Creating a New Base WDAC Policy *(on page 105)*

Deploying a Supplemental Policy *(on page 113)*

Viewing Endpoint Details from Web Reports *(on page 131)*

Post-Migration Administration

Administering BigFix Application Control v2.1.0 *(on page 35)*

BigFix Application Control: Configuration and Installation *(on page 37)*

Provisioning the TPM-Backed Code-Signing Certificate *(on page 44)*

Removing Base and Supplemental Policies *(on page 126)*

Troubleshooting and FAQs

Frequently Asked Questions *(on page 141)*

Troubleshooting Application Control *(on page 156)*

Create and Set Up a Self-Signed Code-Signing Certificate on Endpoint v2.x

This deprecated task generates and configures a self-signed RSA 4096-bit SHA256 code-signing certificate on the endpoint for use with Windows Defender Application Control (WDAC). The task creates required BAC directories, stores the certificate in the `Cert:\LocalMachine\My` store, and exports the public certificate to `BAC\Certs\WDAC_SigningCert.cer` for use in WDAC policy signing and deployment workflows.

Important: This task is deprecated and will be removed from BigFix Application Control v2.x in a future release. Use Provision the TPM-Enforced WDAC Policy Signing Certificate on the WDAC Management Node *(on page 44)* instead.

Use this task to generate and configure a self-signed code-signing certificate on the endpoint for use with Windows Defender Application Control (WDAC).

The task performs the following:

- Creates the following required BAC directories if they do not exist:
  - `BAC\Certs`
  - `BAC\Logs`
- Generates a self-signed code-signing certificate with:

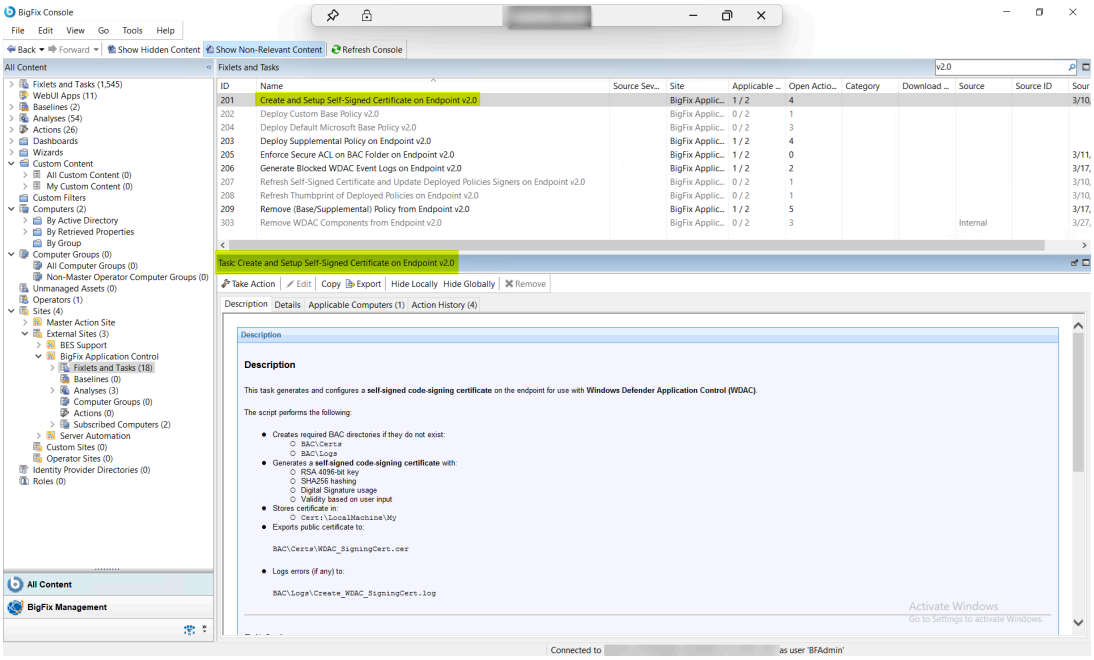
    - RSA 4096-bit key
    - SHA256 hashing
    - Digital Signature usage
    - Validity based on user input
- Stores the certificate at the `Cert:\LocalMachine\My` location.
- Exports the public certificate to the `BAC\Certs\WDAC_SigningCert.cer` folder.
- Logs errors (if any) to the `BAC\Logs\Create_WDAC_SigningCert.log` location.

Refer to the table below to know more about the task's exit code.

Table 30. Create Self-Signed Certificate on Endpoint v2.0: Task Exit Codes and Meanings

Exit Code	Meaning
0	Success
10	BAC directory creation failure
11	Certificate directory creation failure
12	Log directory creation failure
20	Certificate creation failure
30	Certificate export failure

Figure 36. BigFix Console — Create and Set Up Self-Signed Certificate on Endpoint v2.0 Task Overview



1. In the BigFix Console, navigate to **All Content > BigFix Application Control > Fixlets and Tasks**.
2. From the **Fixlets and Tasks** pane, select **Task: Setup and Create Self-Signed Certificate on Endpoint v2.0**.

3. From the **Task: Setup and Create Self-Signed Certificate on Endpoint v2.0** pane, under **Configuration Options**, enter the following information:

   Figure 37. BigFix Console — Setup and Create Self-Signed Certificate on Endpoint v2.0: Configuration Options Panel

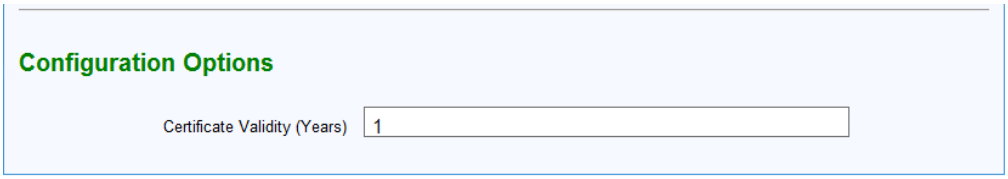
   

   Table 31. Setup and Create Self-Signed Certificate on Endpoint v2.0: Configuration Options Parameter Descriptions

Field Name	Description
Certificate Validity (Years)	Number of years for which the newly generated certificate will be valid.

4. From the **Task: Setup and Create Self-Signed Certificate on Endpoint v2.0** pane, click the **Applicable Computers(n)** tab and view the endpoints on which you want to run the task.
5. Select the **Take Actions** tab and select the endpoints on which you want to apply this installer task.
6. Click **OK**.

A successful execution of this task results in the following outcomes:

- A self-signed WDAC certificate is created on the endpoint.
- The certificate is available in the `LocalMachine` store.
- The public certificate is exported to the `BAC` folder.
- A certificate is ready for WDAC policy signing and deployment.

Current Replacement for This Deprecated Task

Provision the TPM-Enforced WDAC Policy Signing Certificate on the WDAC Management Node *(on page 44)*

Compile, Sign, and Publish WDAC Policies on the WDAC Management Node *(on page 60)*

Deprecated Certificate Management Tasks

Deprecated BigFix Application Control v2.x Fixlets and Tasks *(on page 90)*

Refresh Self-Signed Certificate and Update Deployed Policies Signers on Endpoint v2.0 *(on page 116)*

Refresh Self-Signed Certificate and Update Deployed Policies Signers on Endpoint v2.0 *(on page 116)*

Related Deprecated Policy Deployment Tasks

Deploy Default Microsoft Base Policy v2.0 *(on page 101)*

Deploy Custom Base Policy v2.0 *(on page 109)*

Reference

BigFix Application Control v2.0 Tasks: Exit Codes *(on page 134)*

BigFix Application Control: Troubleshooting WDAC Enforcement Issues *(on page 156)*

Deploy a Default Microsoft WDAC Base Policy on Endpoint v2.x

This deprecated task deploys a signed Windows Defender Application Control (WDAC) base policy on the endpoint using a secure, hash-validated workflow. Select from seven Microsoft-recommended baseline policy templates — including AllowAll, AllowMicrosoft, DefaultWindows (Audit or Enforced), and RecommendedDriverBlock — and the task converts, signs, and deploys the policy to the EFI partition for firmware-level persistence and enforcement across reboots.

Important: This task is deprecated and will be removed from BigFix Application Control v2.x in near future.

This deployment ensures that the policy is trusted, enforced, and persisted at the firmware level by placing it in the EFI partition. This enables strong application control, protecting the system from unauthorized or untrusted code execution.

The task performs the following actions:

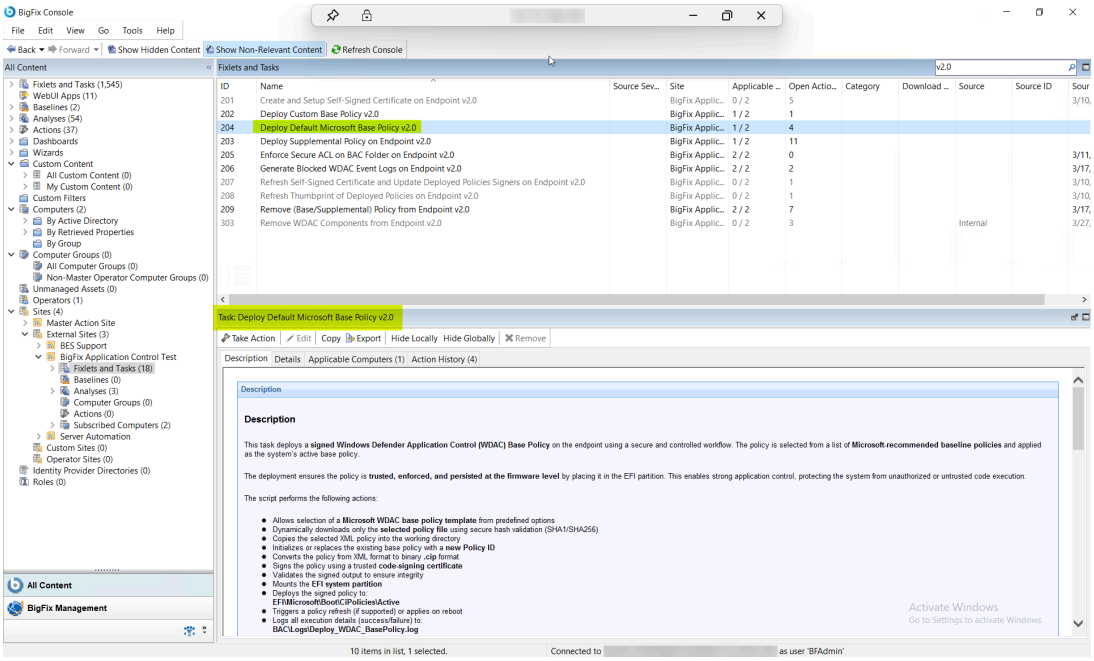
- Allows the selection of a Microsoft WDAC base policy template from the predefined options.
- Dynamically downloads only the selected policy file using secure hash validation (SHA1/SHA256).
- Copies the selected XML policy to the working directory.
- Initializes or replaces the existing base policy with a new policy id.
- Converts the policy from XML format to binary `*.cip` format.
- Signs the policy using a trusted code-signing certificate.
- Validates the signed output to ensure integrity.
- Mounts the EFI system partition.
- Deploys the signed policy to `EFI\Microsoft\Boot\CiPolicies\Active` location.
- Triggers a policy refresh (if supported) or applies on reboot.
- Logs all execution details (success/failure) to `BAC\Logs\Deploy_WDAC_BasePolicy.log` location.

Refer to the table below to know more about the task's exit code.

Table 32. Deploy Default Microsoft Base Policy v2.0: Task Exit Codes and Descriptions

Exit Code	Meaning	Description
0	Success	The WDAC base policy was successfully processed, signed, and deployed.
10	Working directory creation failed	The BAC working directory could not be created or accessed.
11	Log directory creation failed	The logging directory could not be created, preventing execution tracking.
12	Policy directory creation failed	The policy storage directory could not be initialized.
20	Template missing	The selected WDAC policy template file was not found or failed to download.
30	Policy processing failure	Error occurred during policy preparation, version, or rule configuration.
40	Binary conversion failure	Failed to convert the WDAC policy from XML format to binary (.cip).
50	Signing failure	The policy signing process failed or the signed output was not generated correctly.
60	Deployment failure	Failed to mount the EFI partition or copy the policy to the target location.
70	Policy refresh failure	The policy refresh process failed after deployment. A reboot may be required.

Figure 38. BigFix Console — Deploy Default Microsoft Base Policy v2.0 Task Overview



1. In the BigFix Console, navigate to **All Content > BigFix Application Control > Fixlets and Tasks**.
2. From the **Fixlets and Tasks** pane, select **Task: Deploy Default Microsoft Base Policy v2.0**.
3. From the **Task: Deploy Default Microsoft Base Policy v2.0** pane, under **Configuration Options** select one of the following policy options:

Figure 39. BigFix Console — Deploy Default Microsoft Base Policy v2.0: Select Base Policy Configuration Options Panel

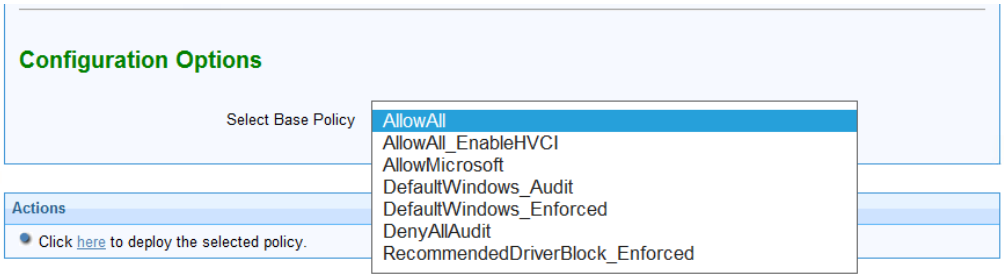


Table 33. Deploy Default Microsoft Base Policy v2.0: Select Base Policy Options and Descriptions

Field Name	Options	Description
Select Base Policy	AllowAll	Permissive policy (allows all applications)
	AllowAll_EnableHVCI	Permissive policy with HVCI support
	AllowMicrosoft	Allows only Microsoft-signed binaries
	DefaultWindows_Audit	Default Microsoft policy in audit mode

Field Name	Options	Description
	DefaultWindows_Enforced	Default Microsoft enforced policy (recommended baseline)
	DenyAllAudit	Blocks all applications in audit mode
	RecommendedDriverBlock_Enforced	Blocks known vulnerable drivers

4. From the **Task: Deploy Default Microsoft Base Policy v2.0** pane, click the **Applicable Computers(n)** tab and view the endpoints on which you want to run the task.
5. Select the **Take Actions** tab and select the endpoints on which you want to apply this installer task.
6. Click **OK**.

A successful execution of this task results in the following outcomes:

- The selected WDAC base policy is successfully deployed to the system.
- The policy becomes the active enforcement policy for Application Control.
- The system enforces or audits applications based on the selected policy type.
- The policy persists across reboots via EFI deployment.
- Execution logs become available for audit and troubleshooting.

Current Replacement for This Deprecated Task

Create and Manage WDAC Policies on the WDAC Management Node *(on page 49)*

Compile, Sign, and Publish WDAC Policies on the WDAC Management Node *(on page 60)*

Deprecated Tasks: Parent Topic and Certificate Prerequisite

Deprecated BigFix Application Control v2.x Fixlets and Tasks *(on page 90)*

Create and Set Up a Self-Signed Code-Signing Certificate on Endpoint v2.0 *(on page 98)*

Deploy Custom Base Policy v2.0 *(on page 109)*

Related Deprecated Policy Management Tasks

Deploy Supplemental Policy on Endpoint v2.0 *(on page 113)*

Remove Base and Supplemental Policy from Endpoint v2.0 *(on page 126)*

Reference

BigFix Application Control v2.0 Tasks: Exit Codes *(on page 134)*

BigFix Application Control: Troubleshooting WDAC Enforcement Issues *(on page 156)*

Create a Custom WDAC Base Policy Using the Policy Wizard

This deprecated topic describes how to use the Microsoft WDAC Policy Wizard to create a custom Windows Defender Application Control (WDAC) base policy. Starting from one of three templates — Default Windows, Allow Microsoft, or Signed and Reputable — configure policy rule options, add custom signing rules (publisher, file path, folder path, or file hash), keep Audit Mode enabled until ready to enforce, then build to generate the policy XML and optional binary for use with the Deploy Custom Base Policy task.

Important: This task is deprecated and will be removed from BigFix Application Control v2.x in near future.

The WDAC Policy Wizard is a tool developed by the Microsoft Windows Defender Application Control (WDAC) feature team to enable IT professionals to create powerful WDAC policies for deployment. Refer to WDAC Policy Wizard to learn more. This document outlines the steps to create a new App Control policy while using one of the three available CI templates as a starting point. The tool enables users to configure the policy rules, its signing rules, and its attributes.

1. **Select the Policy Creator from the Start menu, then Base Policy**

   If the default setting is enabled, the **Base Policy** option will be preselected. Otherwise, select the **Base Policy** option.

2. **Select from one of the default template policies**

   Each of the template policies has a unique set of policy allow list rules that will affect the circle-of-trust and security model of the policy. The following table lists the policies in increasing order of trust and freedom. For instance, the Default Windows mode policy trusts fewer application publishers and signers than the Signed and Reputable mode policy. The Default Windows policy will have a smaller circle-of-trust with better security than the Signed and Reputable policy, but at the expense of compatibility.

   Table 34. WDAC Policy Wizard: Default Template Policy Options and Authorized Components (Increasing Order of Trust)

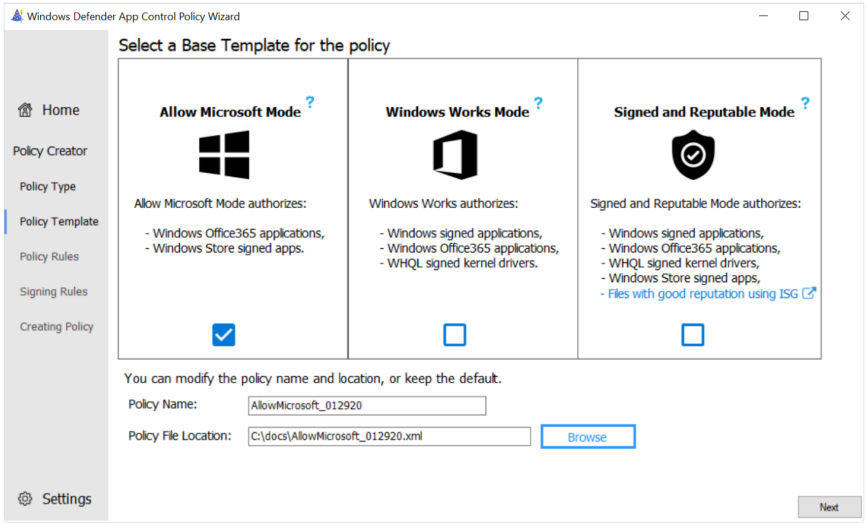
Template Base Policy	Description
Default Windows Mode	Default Windows mode will authorize the following components: ◦ Windows operating components: any binary installed by a fresh install of Windows ◦ Apps installed from the Microsoft Store ◦ Microsoft Office 365 apps, OneDrive, and Microsoft Teams ◦ Third-party Windows Hardware Compatible drivers
Allow Microsoft Mode	Allow mode will authorize the following components: ◦ Windows operating components: any binary installed by a fresh install of Windows ◦ Apps installed from the Microsoft Store

Template Base Policy	Description
	◦ Microsoft Office 365 apps, OneDrive, and Microsoft Teams ◦ Third-party Windows Hardware Compatible drivers ◦ All Microsoft-signed software
Signed and Reputable Mode	Signed and Reputable mode will authorize the following components: ◦ Windows operating components: any binary installed by a fresh install of Windows ◦ Apps installed from the Microsoft Store ◦ Microsoft Office 365 apps, OneDrive, and Microsoft Teams ◦ Third-party Windows Hardware Compatible drivers ◦ All Microsoft-signed software ◦ Files with good reputation per Microsoft Defender's Intelligent Security Graph technology

The policy name and file location will default to values based on the selected template policy. The policy name and file location can be set by selecting the text box and typing the desired string. At any time during the workflow, you can choose to return to the default template page by selecting the **Policy Template** button on the left-hand menu.

Note: Returning to the template page will remove the configured policy rule options as well as the custom signing rules.

Figure 40. WDAC Policy Wizard — Select a Base Template for the Custom Policy



3. **Configure the policy rule options**

Upon page launch, policy rule options will be automatically enabled or disabled depending on the chosen template from the previous page. Choose to enable or disable the desired policy rule options by pressing the slider button next to the policy rule titles.

Hovering the mouse over the policy rule names will display a short description of the rule at the bottom of the page. More information about each of the policy rules can be found on the App Control policy rules page.

Selecting the **+ Advanced Options** button will reveal the advanced policy rule options panel. This grouping of rules contains additional policy rule options that are less common for the majority of users.

Lastly, **Audit Mode** is enabled by default for all of the templates. We recommend leaving the Audit Mode policy rule option enabled until users have fully understood how the policy and signing rules will affect their scenario. Disabling Audit Mode will result in the policy running in enforced mode after the policy is deployed. For more information on deploying App Control policies, see Deploying App Control Policies.

4. **Creating policy signing rules**

   The Signing Rules List on the left-hand side of the page documents the preset signing rules of the template as well as any exceptions.

   Creating Custom Signing Rules

   Selecting the **+ Custom Rules** button will open the **Custom Rules** panel. Four types of custom rule conditions can be defined.

   Table 35. WDAC Policy Wizard: Custom Signing Rule Condition Types and Usage Scenarios

Rule Condition	Usage Scenario
Publisher	To use a publisher condition, the files must be digitally signed by the software publisher, or you must sign with an internal certificate.
File Path	Any file can be assigned this rule condition; however, because path rules specify locations within the file system, any subdirectory will also be affected by the rule (unless explicitly exempted).
Folder Path	Any folder and subfolder can be assigned this rule condition (unless explicitly exempted).
File Hash	Any file can be assigned this rule condition; however, the rule must be updated each time a new version of the file is released because the hash value is based in part on the version.

   Publisher Rules

Select the **Publisher** option from the **Rule Type** combo box. Next, choose to Allow or Deny the publisher, and select a reference file signed by the software publisher on which to base the rule. By default, the publisher is set to apply to all files signed by the publisher, including the specific product name and file name with a version at or above the one specified. The restrictiveness of the rule can be modified using the slider. The text below the slider documents outlines the how the rule will be interpreted.

The table below shows the relationship between the slider placement, the corresponding App Control rule level and its description. The lower the placement on the table and the UI slider, the greater the specificity of the rule.

Table 36. WDAC Policy Wizard Publisher Rule: App Control Rule Levels and Specificity Descriptions (Least to Most Specific)

Rule Condition	App Control Rule Level	Description
Publisher	PCACertificate	Highest available certificate is added to the signers. This is typically the PCA certificate, one level below the root certificate. Any file signed by this certificate will be affected.
Product name	Publisher	This rule is a combination of the PCACertificate rule and the common name (CN) of the leaf certificate. Any file signed by a major CA, but with a leaf certificate from a specific company, for example, a device driver company, is affected.
File name	SignedVersion	This rule is a combination of PCACertificate, Publisher, and a version number. Anything from the specified publisher with a version at or above the one specified is affected.
Version	FilePublisher	Most specific. A combination of the file name, publisher, and PCA certificate as well as a minimum version number. Files from the publisher with the specified name and greater than or equal to the specified version are affected.

Path Rules

Select the **Path** option from the **Rule Type** combo-box. Next, choose to Allow or Deny the path, and select either a File or Folder rule using the radio button below the **Browse** button. Lastly, select the reference file or folder on which to base the rule.

Hash Rules

Select the **File Hash** option from the **Rule Type** combo-box. Next, choose to Allow or Deny the hash, and select the file on which to base the rule.

Deleting Signing Rules

Template signing rules and custom rules can be deleted from the policy by selecting the rule from the rules list data viewer. Once the rule is highlighted, selecting the delete button underneath the table will prompt for additional confirmation. Select **Yes** to remove the rule from the policy and the rules table.

5. **Building the policy**

   The policy build page will monitor the progress of the App Control policy creation process. Depending on the number and complexity of the custom signing rules, the build process could take several minutes.

   This tool creates the custom base policy XML to provide input to the 'Custom Base Policy' task. Refer to Deploy a Custom WDAC Base Policy on Endpoint v2.x *(on page 109)* for more details.

Current Replacement for This Deprecated Workflow

Create and Manage WDAC Policies on the WDAC Management Node *(on page 49)*

Compile, Sign, and Publish WDAC Policies on the WDAC Management Node *(on page 60)*

Deprecated Tasks: Parent Topic and Policy Deployment

Deprecated BigFix Application Control v2.x Fixlets and Tasks *(on page 90)*

Deploy Custom Base Policy v2.0 *(on page 109)*

Deploy Default Microsoft Base Policy v2.0 *(on page 101)*

Related Deprecated Policy and Certificate Tasks

Deploy Supplemental Policy on Endpoint v2.0 *(on page 113)*

Create and Set Up a Self-Signed Code-Signing Certificate on Endpoint v2.0 *(on page 98)*

Reference

BigFix Application Control v2.0 Tasks: Exit Codes *(on page 134)*

BigFix Application Control: Troubleshooting WDAC Enforcement Issues *(on page 156)*

Deploy a Custom WDAC Base Policy on Endpoint v2.x

This deprecated task deploys a custom Windows Defender Application Control (WDAC) base policy on the endpoint from a user-provided XML configuration. The task automatically bumps the policy version if an older version exists, injects mandatory publisher rules for the BigFix Client, Refresh Tool, and SignTool, converts the policy from XML to binary CIP format, signs it with a trusted certificate, and deploys it to the EFI partition for firmware-level persistence and enforcement across reboots.

Important: This task is deprecated and will be removed from BigFix Application Control v2.x in near future.

This deployment ensures that the custom policy is trusted, enforced, and persists at the firmware level by placing it in the EFI partition. The script also automatically handles version increments and injects necessary rules to ensure the BigFix client and deployment tools remain operational.

The task performs the following actions:

- Reads the user-provided Custom XML configuration.
- Dynamically bumps the policy version (if an older version already exists).
- Injects mandatory publisher rules (BigFix Client, Refresh Tool, SignTool).
- Configures rule options (adds 16, 17, 18 and removes 6).
- Converts the policy from XML format to binary `*.cip` format.
- Signs the policy using a trusted code-signing certificate.
- Validates the signed output to ensure integrity.
- Mounts the EFI system partition.
- Deploys the signed policy to `EFI\Microsoft\Boot\CiPolicies\Active` loaction.
- Triggers a policy refresh and safely dismounts the EFI partition.
- Logs all execution details (success/failure) to the `BAC\Logs\Deploy_WDAC_CustomBase.log` location.

Refer to the table below to know more about the task's exit code.

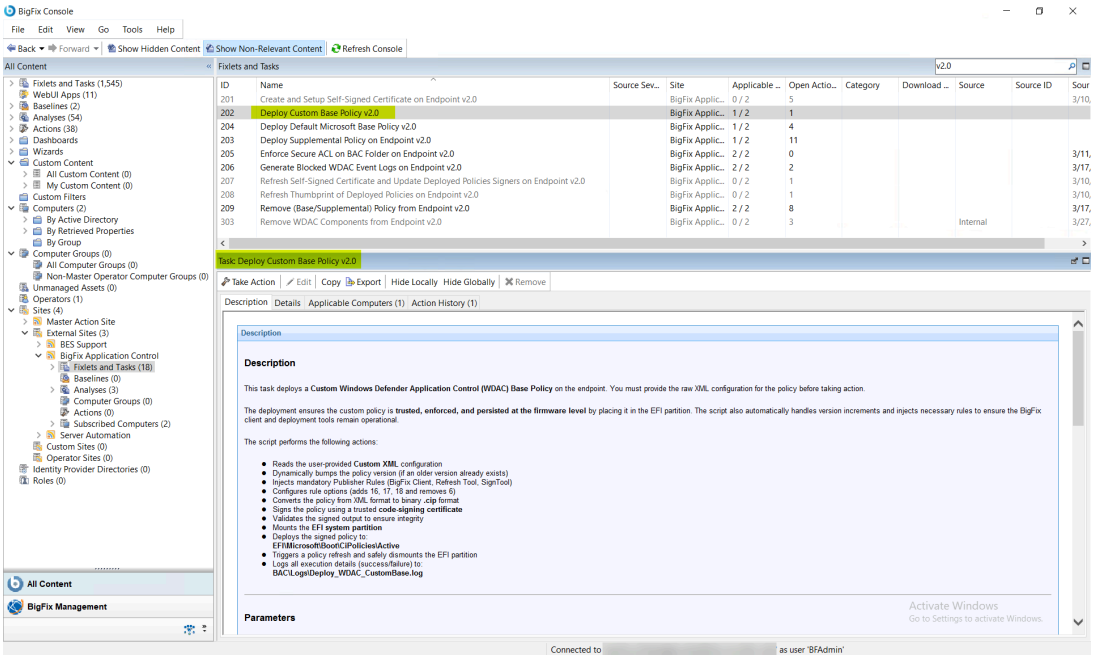
Table 37. Deploy Custom Base Policy v2.0: Task Exit Codes and Descriptions

Exit Code	Meaning	Description
0	Success	The WDAC base policy was successfully processed, signed, and deployed.
10	Working directory creation failed	The BAC working directory could not be created or accessed.
11	Log directory creation failed	The logging directory could not be created, preventing execution tracking.
12	Policy directory creation failed	The policy storage directory could not be initialized.
30	Policy processing failure	Error occurred during policy preparation, version bumping, or rule injection.
40	Binary conversion failure	Failed to convert the WDAC policy from XML format to binary (.cip).
50	Signing failure	The policy signing process failed or generated an invalid signature.

Table 37. Deploy Custom Base Policy v2.0: Task Exit Codes and Descriptions (continued)

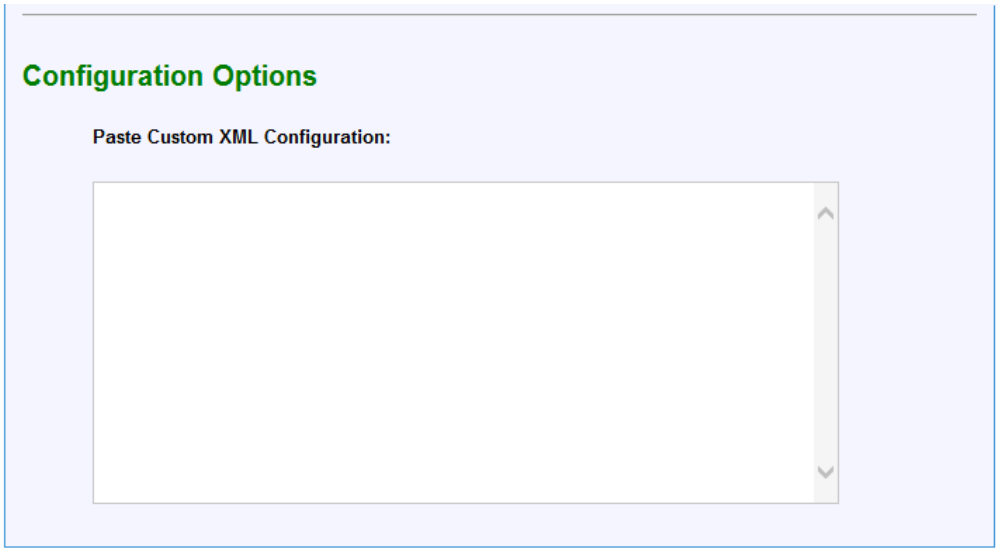
Exit Code	Meaning	Description
60	Deployment failure	Failed to mount the EFI partition or copy the policy to the target location.
70	Policy refresh failure	The policy refresh process or EFI dismount failed.

Figure 41. BigFix Console — Deploy Custom Base Policy v2.0 Task Overview



1. In the BigFix Console, navigate to **All Content > BigFix Application Control > Fixlets and Tasks**.
2. From the **Fixlets and Tasks** pane, select **Task: Deploy Custom Base Policy v2.0**.
3. From the **Task: Deploy Custom Base Policy v2.0** pane, under **Configuration Options**, paste the raw XML configuration for your base policy into the provided text area.

Figure 42. BigFix Console — Deploy Custom Base Policy v2.0: Configuration Options Panel Showing Custom XML Input Text Area



4. From the **Task: Deploy Custom Base Policy v2.0** pane, click the **Applicable Computers(n)** tab and view the endpoints on which you want to run the task.
5. Select the **Take Actions** tab and select the endpoints on which you want to apply this installer task.
6. Click **OK**.

A successful execution of this task results in the following outcomes:

- The custom WDAC base policy is successfully deployed to the system.
- The system enforces or audits applications based on the provided XML logic.
- The policy persists across reboots via EFI deployment.
- Execution logs become available for audit and troubleshooting.

Current Replacement for This Deprecated Task

Create and Manage WDAC Policies on the WDAC Management Node *(on page 49)*

Compile, Sign, and Publish WDAC Policies on the WDAC Management Node *(on page 60)*

Deprecated Tasks: Parent Topic and Policy Authoring Prerequisite

Deprecated BigFix Application Control v2.x Fixlets and Tasks *(on page 90)*

Create a Custom WDAC Base Policy Using the Policy Wizard *(on page 105)*

Deploy Default Microsoft Base Policy v2.0 *(on page 101)*

Related Deprecated Certificate and Policy Tasks

Create and Set Up a Self-Signed Code-Signing Certificate on Endpoint v2.0 *(on page 98)*

Deploy Supplemental Policy on Endpoint v2.0 *(on page 113)*

Reference

BigFix Application Control v2.0 Tasks: Exit Codes *(on page 134)*

BigFix Application Control: Troubleshooting WDAC Enforcement Issues *(on page 156)*

Deploy a Supplemental WDAC Policy on Endpoint v2.x

This deprecated task creates and deploys a signed Windows Defender Application Control (WDAC) supplemental policy on the endpoint. Accept rule input in Single-Level or Multi-Level CSV format, generate WDAC rules from file hash, publisher, file path, or file name attributes, link the new supplemental policy to an existing base policy by GUID, and deploy it signed to the EFI partition in Enforced or Audit mode. Each execution creates a new unique supplemental policy without modifying existing policies.

Important: This task is deprecated and will be removed from BigFix Application Control v2.x in near future.

The task performs the following actions:

- Accepts the rule input in either **Single-Level** or **Multi-Level** (CSV) format.
- Generates WDAC rules based on file attributes such as hash, publisher, file path or file name.
- Creates a new supplemental policy using multiple policy format.
- Links the newly created supplemental policy to the specified base policy GUID.
- Supports enforced and audit execution modes.
- Converts the policy from XML to binary `*.cip` format.
- Signs the policy using a trusted code-signing certificate.
- Mounts the EFI system partition to deploy the policy.
- Copies the signed policy to the `EFI\Microsoft\Boot\CiPolicies\Active` location.
- Ensures each execution creates a new unique supplemental policy without modifying existing policies.
- Logs execution details (success/failure) to the `BAC\Logs\Deploy_WDAC_SuppPolicy.log` location.

Refer to the table below to know more about the task's exit code.

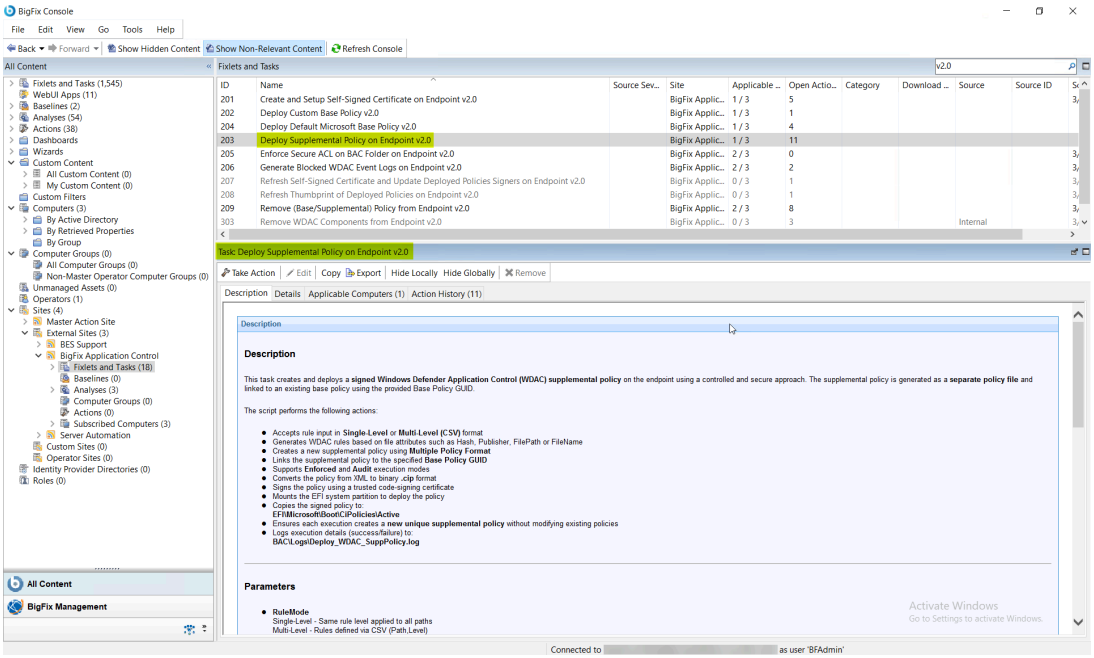
Table 38. Deploy Supplemental Policy on Endpoint v2.0: Task Exit Codes and Meanings

Exit Code	Meaning
0	Success
20	Missing base policy GUID
30	No valid rules generated

Table 38. Deploy Supplemental Policy on Endpoint v2.0: Task Exit Codes and Meanings (continued)

Exit Code	Meaning
40	Signing failure or invalid signed output

Figure 43. BigFix Console — Deploy Supplemental WDAC Policy on Endpoint v2.0 Task Overview



1. In the BigFix Console, navigate to **All Content > BigFix Application Control > Fixlets and Tasks**.
2. From the **Fixlets and Tasks** pane, select **Task: Deploy Supplemental Policy on Endpoint v2.0**.
3. From the **Task: Deploy Supplemental Policy on Endpoint v2.0** pane, under **Configuration Options**, enter the supplemental policy configuration using the fields described below.

Figure 44. BigFix Console — Deploy Supplemental Policy on Endpoint v2.0: Configuration Options Panel

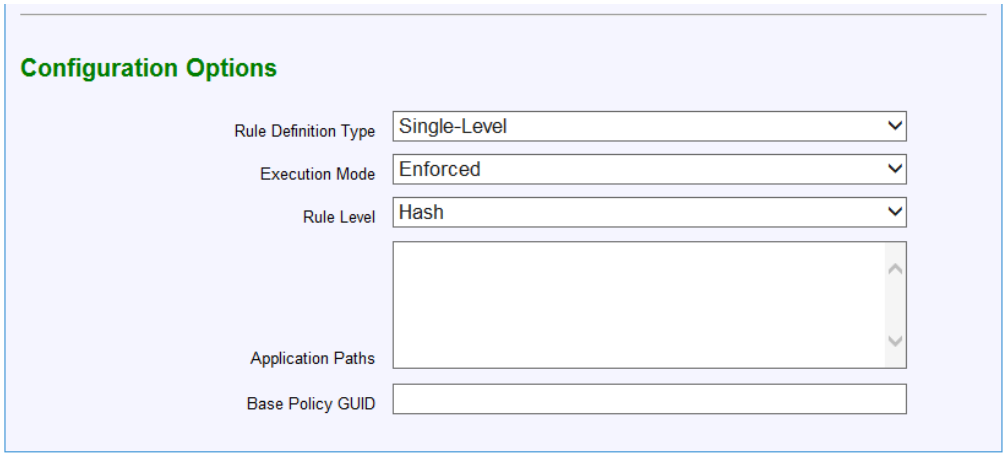


Table 39. Deploy Supplemental Policy on Endpoint v2.0: Configuration Options Parameter Descriptions

Field Name	Options	Description
Rule Definition Type	Single-Level	Same rule level applied to all paths
	Multi-Level	Rules defined via CSV (path, level)
Execution Mode	Enforced	Policy is enforced
	Audit	Policy is applied in audit mode
Rule Level	Hash	Applicable only in Single-Level mode (Hash, Publisher, FilePath, FileName)
	Publisher	
	FilePath	
	FileName	
Application Paths	N/A	Text box. List of file paths (newline or comma separated) for Single-Level mode only
CSV Rules	N/A	Text box. CSV input for Multi-Level mode in the format: Path, Rule Level
Base Policy GUID	N/A	Text field. Required. Specifies the GUID of the base policy to be supplemented.

4. From the **Task: Deploy Supplemental Policy on Endpoint v2.0** pane, click the **Applicable Computers(n)** tab and view the endpoints on which you want to run the task.
5. Select the **Take Actions** tab and select the endpoints on which you want to apply this installer task.
6. Click **OK**.

A successful execution of this task results in the following outcomes:

- A new WDAC supplemental policy is created and deployed.
- The policy is linked to the specified base policy.
- Existing policies remain unchanged.
- The endpoint enforces or audits rules based on the selected mode.

Current Replacement for This Deprecated Task

Create and Manage WDAC Policies on the WDAC Management Node *(on page 49)*

Compile, Sign, and Publish WDAC Policies on the WDAC Management Node *(on page 60)*

Deprecated Tasks: Parent Topic and Base Policy Prerequisites

Deprecated BigFix Application Control v2.x Fixlets and Tasks *(on page 90)*

Deploy Default Microsoft Base Policy v2.0 *(on page 101)*

Deploy Custom Base Policy v2.0 *(on page 109)*

Related Deprecated Certificate and Policy Removal Tasks

Create and Set Up a Self-Signed Code-Signing Certificate on Endpoint v2.0 *(on page 98)*

Remove Base and Supplemental Policy from Endpoint v2.0 *(on page 126)*

Reference

BigFix Application Control v2.0 Tasks: Exit Codes *(on page 134)*

BigFix Application Control: Troubleshooting WDAC Enforcement Issues *(on page 156)*

Refresh the Self-Signed Certificate and Update Deployed WDAC Policies Signers on Endpoint v2.x

This deprecated task automates the end-to-end renewal, signing, and redeployment of Windows Defender Application Control (WDAC) policies on the endpoint. The task checks whether the current self-signed WDAC signing certificate expires within 30 days and, if so, generates a new 4096-bit RSA replacement. It then scans local XML policies, auto-increments their version numbers, updates signer rules to trust the new certificate, converts them to binary CIP format, and redeploys to the EFI partition — all in a single automated workflow.

Important: This task is deprecated and will be removed from BigFix Application Control v2.x in near future.

The task follows the below listed workflow:

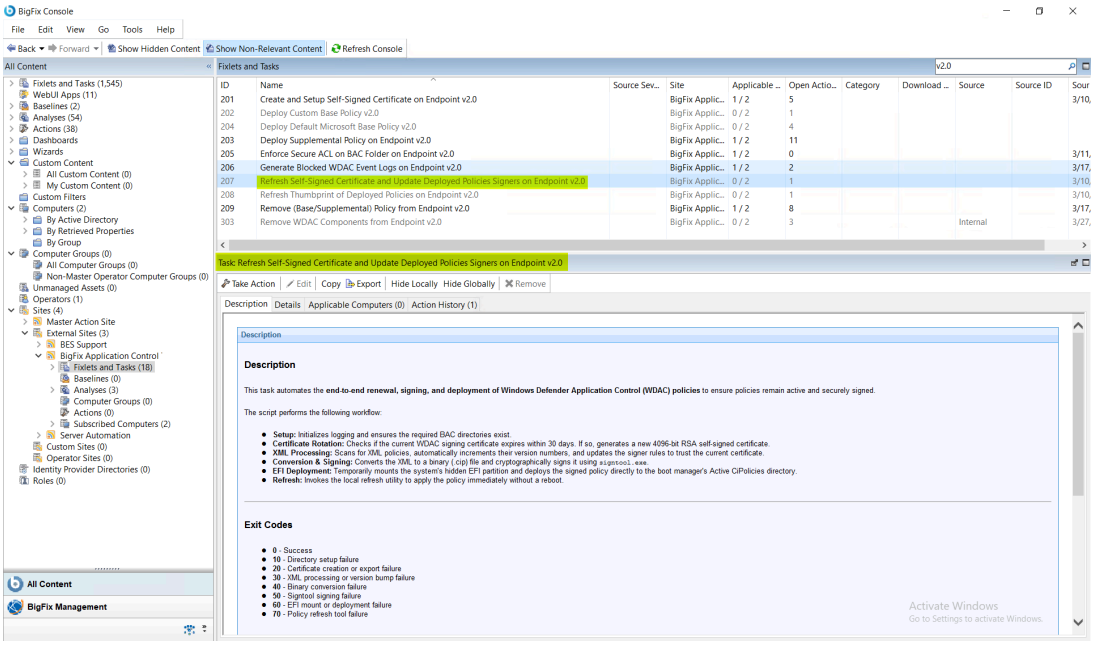
1. **Setup**: Initializes logging and ensures that the required BAC directories exist.
2. **Certificate Rotation**: Checks if the current WDAC signing certificate expires within 30 days. If yes, this task generates a new 4096-bit RSA self-signed certificate.
3. **XML Processing**: Scans for XML policies, automatically increments their version numbers, and updates the signer rules to trust the current certificate.
4. **Conversion & Signing**: Converts the XML to a binary `*.cip` file and cryptographically signs it using the `signtool.exe`.
5. **EFI Deployment**: Temporarily mounts the system's hidden EFI partition and deploys the signed policy directly to the boot manager's `Active CiPolicies` directory.
6. **Refresh**: Invokes the local refresh utility to apply the policy immediately without a reboot.

Refer to the table below to know more about the task's exit code.

Table 40. Refresh Self-Signed Certificate and Update Deployed Policies Signers v2.0: Task Exit Codes and Meanings

Exit Code	Meaning
0	Success
10	Directory setup failure
20	Certificate creation or export failure
30	XML processing or version bumping failure
40	Binary conversion failure
50	Signtool signing failure
60	EFI mount or deployment failure
70	Policy refresh tool failure

Figure 45. BigFix Console — Refresh Self-Signed Certificate and Update Deployed WDAC Policies Signers on Endpoint v2.0 Task Overview



1. In the BigFix Console, navigate to **All Content > BigFix Application Control > Fixlets and Tasks**.
2. From the **Fixlets and Tasks** pane, select **Task: Refresh Self-Signed Certificate and Update Deployed Policies Signers on Endpoint v2.0**.
3. From the **Task: Refresh Self-Signed Certificate and Update Deployed Policies Signers on Endpoint v2.0** pane, under **Configuration Options**, enter the following information:

Figure 46. BigFix Console — Refresh Self-Signed Certificate v2.0: Configuration Options Panel Showing Certificate Validity Years Input

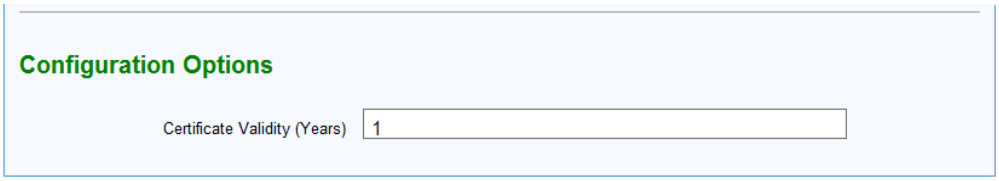


Table 41. Refresh Self-Signed Certificate and Update Deployed Policies Signers v2.0: Configuration Options Parameter Descriptions

Field Name	Description
Certificate Validity (Years)	Number of years for which the newly generated certificate will be valid.

4. From the **Task: Refresh Self-Signed Certificate and Update Deployed Policies Signers on Endpoint v2.0** pane, click the **Applicable Computers(n)** tab and view the endpoints on which you want to run the task.
5. Select the **Take Actions** tab and select the endpoints on which you want to apply this installer task.
6. Click **OK**.

Note: Once this task is triggered, it remains as an open action and runs on the system whenever the system becomes relevant. Only an administrator can stop this task.

A successful execution of this task results in the following outcomes:

- WDAC signing certificate is rotated (if within 30 days of expiration).
- Local XML policies are updated with incremented version numbers and new signer rules.
- Cryptographically signed binary (*.cip) policies are successfully deployed to the EFI partition.
- System application control policies are immediately refreshed and enforced.
- A system reboot is required for the changes to take effect.

Current Replacement for This Deprecated Task

Provision the TPM-Enforced WDAC Policy Signing Certificate on the WDAC Management Node *(on page 44)*

Compile, Sign, and Publish WDAC Policies on the WDAC Management Node *(on page 60)*

Deprecated Tasks: Parent Topic and Related Certificate Tasks

Deprecated BigFix Application Control v2.x Fixlets and Tasks *(on page 90)*

Create and Set Up a Self-Signed Code-Signing Certificate on Endpoint v2.0 *(on page 98)*

Refresh Thumbprint of Deployed Policies Signers on Endpoint v2.0 *(on page 119)*

Related Deprecated Policy Deployment Tasks

Deploy Default Microsoft Base Policy v2.0 *(on page 101)*

Deploy Custom Base Policy v2.0 *(on page 109)*

Reference

BigFix Application Control v2.0 Tasks: Exit Codes *(on page 134)*

BigFix Application Control: Troubleshooting WDAC Enforcement Issues *(on page 156)*

Refresh the Thumbprint of Deployed WDAC Policies Signers on Endpoint v2.x

This deprecated task updates the code-signing certificate thumbprint across all deployed Windows Defender Application Control (WDAC) policies on the endpoint without generating a new certificate. It scans local XML policies, increments their version numbers, updates signer rules to reference the current certificate thumbprint, converts them to binary CIP format using signtool.exe, and redeploys to the EFI partition — refreshing policy enforcement immediately in a single automated five-step workflow.

Important: This task is deprecated and will be removed from BigFix Application Control v2.x in near future.

The task follows the below listed workflow:

1. **Setup**: Initializes the logging and ensures that the required BAC directories exist.
2. **XML Processing**: Scans for the XML policies, automatically increments their version numbers, and updates the signer rules to trust the current certificate.
3. **Conversion & Signing**: Converts the XML to a binary (`*.cip`) file and cryptographically signs it using `signtool.exe`.
4. **EFI Deployment**: Temporarily mounts the system's hidden EFI partition and deploys the signed policy directly to the boot manager's `Active CiPolicies` directory.
5. **Refresh**: Invokes the local refresh utility to apply the policy immediately without a reboot.

Refer to the table below to know more about the task's exit code.

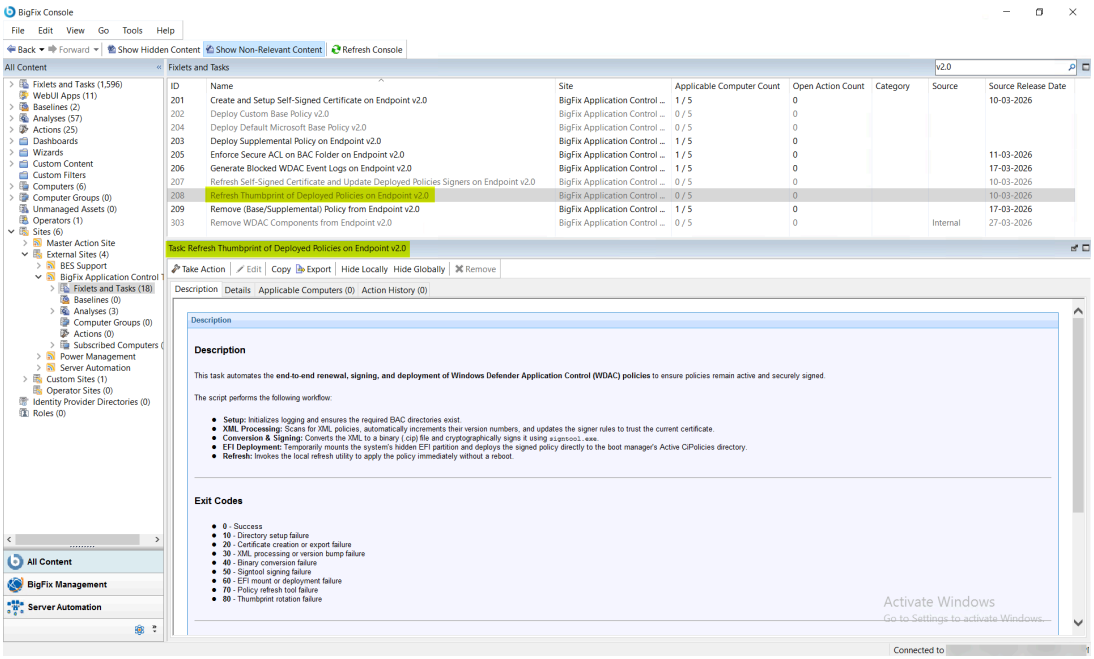
Table 42. Refresh Thumbprint of Deployed WDAC Policies Signers v2.0: Task Exit Codes and Meanings

Exit Code	Meaning
0	Success
10	Directory setup failure
20	Certificate creation or export failure
30	XML processing or version bumping failure
40	Binary conversion failure

Table 42. Refresh Thumbprint of Deployed WDAC Policies Signers v2.0: Task Exit Codes and Meanings (continued)

Exit Code	Meaning
50	Signtool signing failure
60	EFI mount or deployment failure
70	Policy refresh tool failure
80	Thumbprint rotation failure

Figure 47. BigFix Console — Refresh Thumbprint of Deployed WDAC Policies Signers on Endpoint v2.0 Task Overview



1. In the BigFix Console, navigate to **All Content > BigFix Application Control > Fixlets and Tasks**.
2. From the **Fixlets and Tasks** pane, select **Task: Refresh Thumbprint of Deployed Policies Signers on Endpoint v2.0**.
3. From the **Task: Refresh Thumbprint of Deployed Policies Signers on Endpoint v2.0** pane, click the **Applicable Computers(n)** tab and view the endpoints on which you want to run the task.
4. Select the **Take Actions** tab and select the endpoints on which you want to apply this installer task.
5. Click **OK**.

Note: Once this task is triggered, it remains as an open action and runs on the system whenever the system becomes relevant. Only an administrator can stop this task.

A successful execution of this task results in the following outcomes:

- Local XML policies are updated with incremented version numbers and new signer rules.
- Cryptographically signed binary (*.cip) policies are successfully deployed to the EFI partition.
- System application control policies are immediately refreshed and enforced.

Current Replacement for This Deprecated Task

Provision the TPM-Enforced WDAC Policy Signing Certificate on the WDAC Management Node *(on page 44)*

Compile, Sign, and Publish WDAC Policies on the WDAC Management Node *(on page 60)*

Deprecated Tasks: Parent Topic and Related Certificate Tasks

Deprecated BigFix Application Control v2.x Fixlets and Tasks *(on page 90)*

Create and Set Up a Self-Signed Code-Signing Certificate on Endpoint v2.0 *(on page 98)*

Refresh the Self-Signed Certificate and Update Deployed WDAC Policies Signers on Endpoint v2.0 *(on page 116)*

Related Deprecated Policy Deployment Tasks

Deploy Default Microsoft Base Policy v2.0 *(on page 101)*

Deploy Custom Base Policy v2.0 *(on page 109)*

Reference

BigFix Application Control v2.0 Tasks: Exit Codes *(on page 134)*

BigFix Application Control: Troubleshooting WDAC Enforcement Issues *(on page 156)*

Enforce Secure ACL Permissions on the BAC Folder on Endpoint v2.x

This deprecated task hardens the BigFix Application Control (BAC) directory at `\Program Files (x86)\BigFix Enterprise\BES Client\BAC` by enforcing strict Windows Access Control List (ACL) settings: it disables ACL inheritance, removes all existing inherited permissions, grants exclusive full control to `NT AUTHORITY\SYSTEM`, and recursively sets SYSTEM as the owner of all files and subfolders — protecting WDAC policy artifacts, signing certificates, and execution logs from unauthorized access.

Important: This task is deprecated and will be removed from BigFix Application Control v2.x in near future.

The task performs the following actions:

- Validates the existence of the BAC directory at the `\Program Files (x86)\BigFix Enterprise\BES Client\BAC` location. If the directory is missing, the task exits with a failure code.
- Ensures the logging directory exists and writes execution logs to the `BAC\Logs \Secure_BAC_Permissions.log` location.
- Updates Access Control Lists (ACLs):

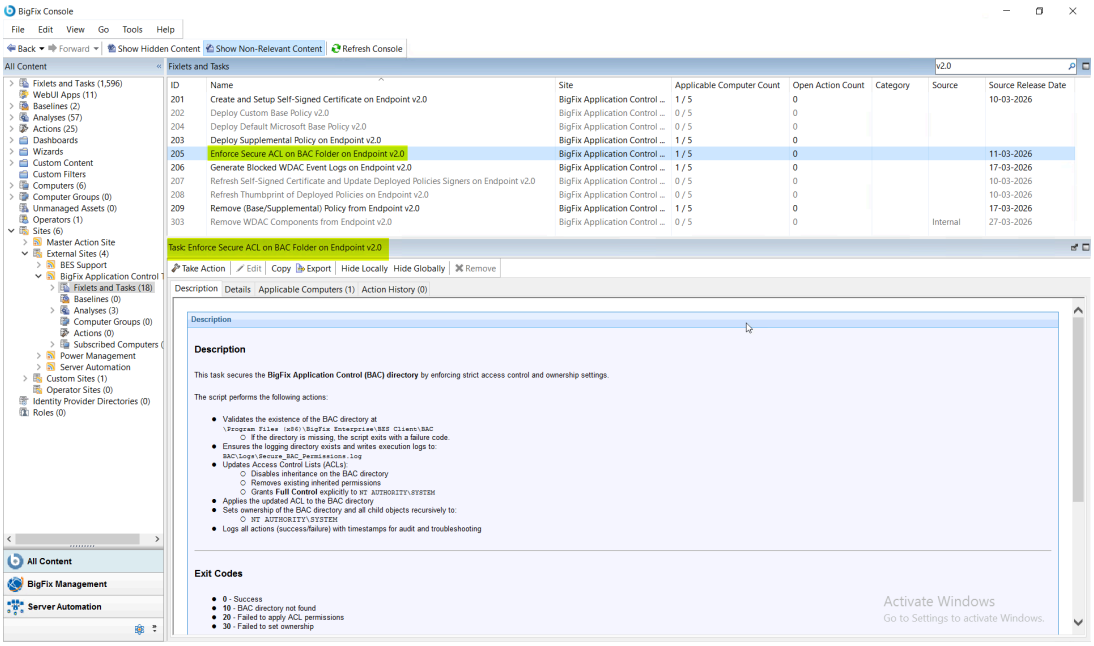
    - Disables inheritance on the BAC directory.
    - Removes any existing inherited permissions.
    - Grants full control explicitly to `NT AUTHORITY\SYSTEM`.
- Applies the updated ACL to the BAC directory.
- Sets ownership of the BAC directory and all child objects recursively to `NT AUTHORITY\SYSTEM`.
- Logs all actions (success/failure) with timestamps for audit and troubleshooting.

Refer to the table below to know more about the task's exit code.

Table 43. Enforce Secure ACL on BAC Folder v2.0: Task Exit Codes and Meanings

Exit Code	Meaning
0	Success
10	BAC directory not found
20	Failed to apply ACL permissions
30	Failed to set ownership

Figure 48. BigFix Console — Enforce Secure ACL Permissions on the BAC Folder on Endpoint v2.0 Task Overview



1. In the BigFix Console, navigate to **All Content > BigFix Application Control > Fixlets and Tasks**.
2. From the **Fixlets and Tasks** pane, select **Task: Enforce Secure ACL on BAC Folder on Endpoint v2.0**.
3. From the **Task: Enforce Secure ACL on BAC Folder on Endpoint v2.0** pane, click the **Applicable Computers(n)** tab and view the endpoints on which you want to run the task.
4. Select the **Take Actions** tab and select the endpoints on which you want to apply this installer task.

5. Click **OK**.

Note: Once this task is triggered, it remains as an open action and runs on the system whenever the system becomes relevant. Only an administrator can stop this task.

A successful execution of this task results in the following outcomes:

- The BAC directory is fully secured.
- Only the SYSTEM account has full control.
- Ownership is standardized across all files and sub folders.
- Unauthorized or inherited permissions are removed.

BAC Installation and Administration

Install and Configure the BigFix Application Control (BAC) Component *(on page 37)*

Administrating BigFix Application Control v2.1.0 *(on page 35)*

Deprecated Tasks: Parent Topic and Endpoint Cleanup Tasks

Deprecated BigFix Application Control v2.x Fixlets and Tasks *(on page 90)*

Uninstall WDAC Components from Target Endpoint *(on page 75)*

Uninstall BAC Configuration from Root Server *(on page 70)*

Related Deprecated Tasks: Certificate and Policy Management

Create and Set Up a Self-Signed Code-Signing Certificate on Endpoint v2.0 *(on page 98)*

Refresh the Self-Signed Certificate and Update Deployed WDAC Policies Signers on Endpoint v2.0 *(on page 116)*

Reference

BigFix Application Control v2.0 Tasks: Exit Codes *(on page 134)*

BigFix Application Control: Troubleshooting WDAC Enforcement Issues *(on page 156)*

Generate Blocked WDAC Event Logs on Endpoint v2.x

This deprecated task extracts Windows Defender Application Control (WDAC) block events from two Windows event log sources — `Microsoft-Windows-CodeIntegrity/Operational` and `Microsoft-Windows-AppLocker/MSI and Script` — for a configurable timeframe of 1 to 14 days. It filters audit-mode event IDs 3076, 3034, and 8028 ("would be blocked") and enforced-mode event IDs 3077, 3033, 8029, and 8040 ("blocked"), then outputs a structured JSON report to `BAC\WDAC_Reports\WDAC_Blocked_Events.json` for dashboard ingestion, policy analysis, or BigFix Web Reports.

Important: This task is deprecated and will be removed from BigFix Application Control v2.x in near future.

The task performs the following actions:

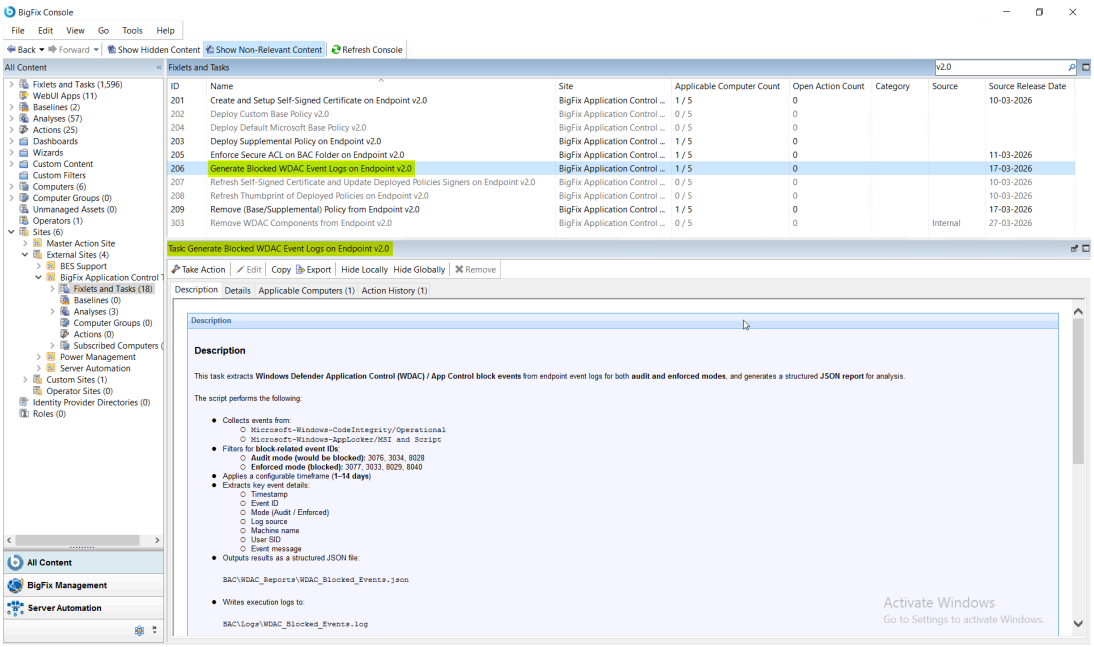
- Collects events from:
  - `Microsoft-Windows-CodeIntegrity/Operational`
  - `Microsoft-Windows-AppLocker/MSI and Script`
- Filters for block-related event IDs:
  - Audit mode (would be blocked): 3076, 3034, 8028
  - Enforced mode (blocked): 3077, 3033, 8029, 8040
- Applies a configurable time frame of 1 to 14 days.
- Extracts the following key event details:
  - Timestamp
  - Event ID
  - Mode (Audit / Enforced)
  - Log source
  - Machine name
  - User SID
  - Event message
- Outputs the results as a structured JSON file to `BAC\WDAC_Reports\WDAC_Blocked_Events.json` file.
- Writes the execution logs to the `BAC\Logs\WDAC_Blocked_Events.log` file.

Refer to the table below to know more about the task's exit code.

Table 44. Generate Blocked WDAC Event Logs v2.0: Task Exit Codes and Meanings

Exit Code	Meaning
0	Success
10	Invalid time frame
20	BAC path not found
30	Event query failure
40	No events found
50	Report generation failure

Figure 49. BigFix Console — Generate Blocked WDAC Event Logs on Endpoint v2.0 Task Overview



1. In the BigFix Console, navigate to **All Content > BigFix Application Control > Fixlets and Tasks**.
2. From the **Fixlets and Tasks** pane, select **Task: Generate Blocked WDAC Event Logs on Endpoint v2.0**.
3. From the **Task: Generate Blocked WDAC Event Logs on Endpoint v2.0** pane, under **Configuration Options**, enter the following information:

   Figure 50. BigFix Console — Generate Blocked WDAC Event Logs v2.0: Configuration Options Panel Showing Timeframe Days Input

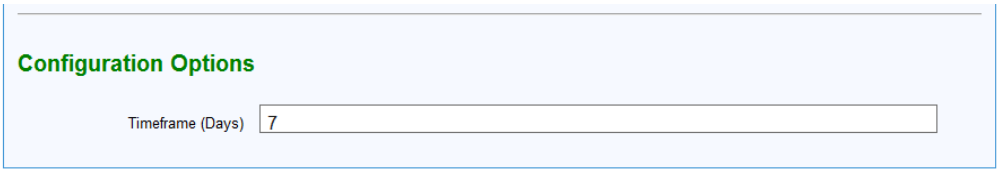
   

   Table 45. Generate Blocked WDAC Event Logs v2.0: Configuration Options Parameter Descriptions

Field Name	Description
Timeframe (Days)	Number of days for which the event logs will be generated.

4. From the **Task: Generate Blocked WDAC Event Logs on Endpoint v2.0** pane, click the **Applicable Computers(n)** tab and view the endpoints on which you want to run the task.
5. Select the **Take Actions** tab and select the endpoints on which you want to apply this installer task.
6. Click **OK**.

A successful execution of this task results in the following outcomes:

- WDAC block events, both audit and enforced are extracted.
- Clean, structured JSON report is generated in the BAC folder.
- Data ready for ingestion into dashboards, analysis, or for BigFix® reporting.

WDAC Reporting and Policy Review

View Endpoint Details from BigFix Web Reports *(on page 131)*

Submit a WDAC Policy for Review *(on page 79)*

Related Deprecated Policy Deployment Tasks

Deploy Default Microsoft Base Policy v2.0 *(on page 101)*

Deploy Custom Base Policy v2.0 *(on page 109)*

Deploy Supplemental Policy v2.0 *(on page 113)*

Target Endpoint WDAC Setup and Policy Management

Set Up and Initialize WDAC on Target Endpoint *(on page 64)*

Deploy and Refresh WDAC Policy on Target Endpoint *(on page 66)*

Reference

BigFix Application Control v2.0 Tasks: Exit Codes *(on page 134)*

BigFix Application Control: Troubleshooting WDAC Enforcement Issues *(on page 156)*

Remove Base or Supplemental WDAC Policy from Endpoint v2.x

This deprecated task removes Windows Defender Application Control (WDAC) policies from the endpoint in one of two modes: **Supplemental mode** removes a single targeted supplemental policy identified by its Policy GUID from both EFI and System32 storage locations; **Base mode** performs a full reset by removing all existing WDAC base and supplemental policies from both locations and deploying a clean `AllowAll.cip` policy to prevent application blocking. A system reboot is required for changes to take effect.

Important: This task is deprecated and will be removed from BigFix Application Control v2.x in near future.

The task performs the following actions:

- Mounts the EFI system partition to access the boot-level WDAC policies.
- Supports the following two operational modes:
  - **Supplemental Mode**: Removes a specific WDAC supplemental policy using the provided Policy GUID.
  - **Base Mode**:

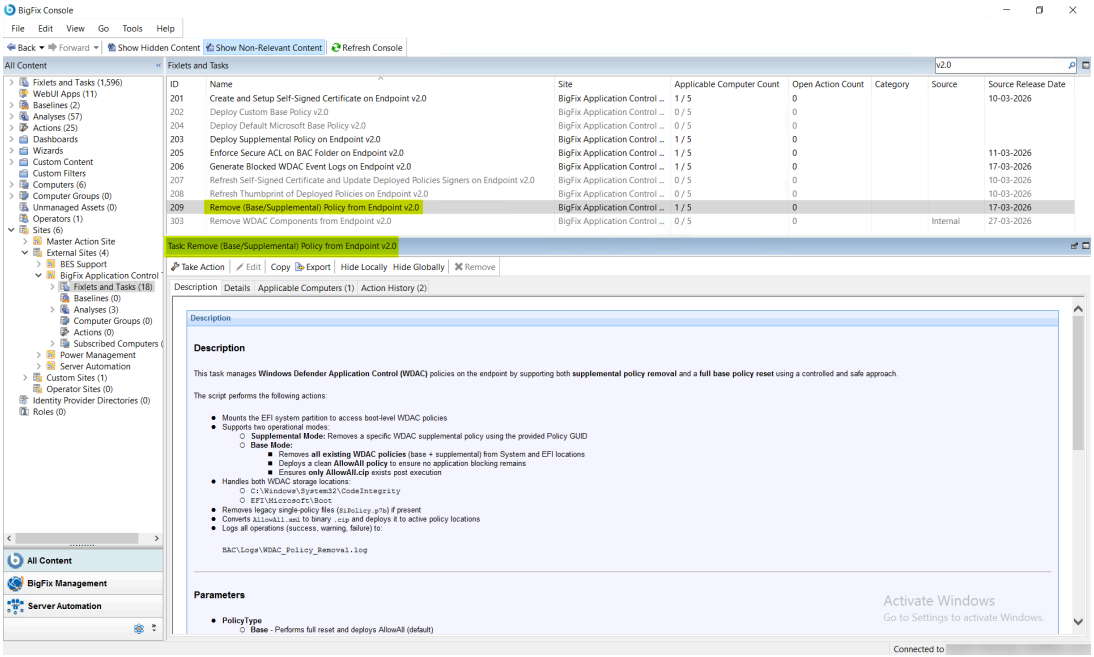
        - Removes all the existing WDAC policies (base as well as supplemental) from System and EFI locations.
        - Deploys a clean `AllowAll` policy to ensure no application blocking exists.
        - Ensures only `AllowAll.cip` exists post this task execution.
    - Handles both the WDAC storage locations:
        - `C:\Windows\System32\CodeIntegrity`
        - `EFI\Microsoft\Boot`
    - Removes legacy single-policy files (like `SiPolicy.p7b`) if present.
    - Converts the `AllowAll.xml` to binary `*.cip` and deploys it to the active policy locations.
    - Logs all the operations (success, warning, failure) in the `BAC\Logs\WDAC_Policy_Removal.log` file.

Refer to the table below to know more about the task's exit code.

Table 46. Remove Base or Supplemental WDAC Policy v2.0: Task Exit Codes and Meanings

Exit Code	Meaning
0	Success
10	Invalid input parameters
20	BAC path not found
30	EFI mount failure
40	Policy removal failure
50	AllowAll.xml not found
60	AllowAll deployment failure

Figure 51. BigFix Console — Remove Base or Supplemental WDAC Policy from Endpoint v2.0 Task Overview



1. In the BigFix Console, navigate to **All Content > BigFix Application Control > Fixlets and Tasks**.
2. From the **Fixlets and Tasks** pane, select **Task: Remove (Base/Supplemental) Policy from Endpoint v2.0**.
3. From the **Task: Remove (Base/Supplemental) Policy from Endpoint v2.0** pane, under **Configuration Options**, enter the following information:

Figure 52. BigFix Console — Remove Base or Supplemental WDAC Policy v2.0: Configuration Options Panel Showing Policy Type and PolicyId Input Fields

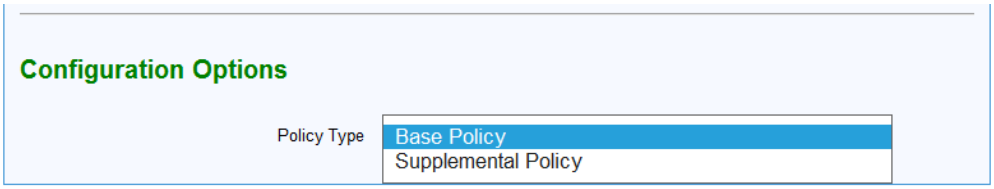


Table 47. Remove Base or Supplemental WDAC Policy v2.0: Configuration Options Parameter Descriptions

Field Name	Options	Description
Policy Type	Base	Performs full reset and deploys AllowAll (default).
	Supplemental	Removes only a specific supplemental policy.
PolicyId		Required only when the policy type is Supplemental. Specifies

Field Name	Options	Description
		the GUID of the policy to be removed.

4. From the **Task: Remove (Base/Supplemental) Policy from Endpoint v2.0** pane, click the **Applicable Computers(n)** tab and view the endpoints on which you want to run the task.
5. Select the **Take Actions** tab and select the endpoints on which you want to apply this installer task.
6. Click **OK**.

A successful execution of this task results in the following outcomes:

- In Supplemental mode, only the specified policy is removed.
- In Base mode, all WDAC policies are removed and replaced with AllowAll.
- System remains stable with a valid WDAC configuration.
- A system reboot is required for changes to take effect.

Current WDAC Policy Management

Manage WDAC Policies on the WDAC Management Node *(on page 49)*

Deploy and Refresh WDAC Policy on Target Endpoint *(on page 66)*

Deprecated Tasks: Parent Topic and Full WDAC Removal

Deprecated BigFix Application Control v2.x Fixlets and Tasks *(on page 90)*

Remove WDAC from Endpoint v2.0 *(on page 129)*

Uninstall WDAC Components from Target Endpoint *(on page 75)*

Related Deprecated Policy Deployment Tasks

Deploy Default Microsoft Base Policy v2.0 *(on page 101)*

Deploy Custom Base Policy v2.0 *(on page 109)*

Deploy Supplemental Policy v2.0 *(on page 113)*

Reference

BigFix Application Control v2.0 Tasks: Exit Codes *(on page 134)*

BigFix Application Control: Troubleshooting WDAC Enforcement Issues *(on page 156)*

Remove WDAC Components from Endpoint v2.x

This deprecated task performs a complete WDAC infrastructure cleanup on the endpoint: it mounts the EFI system partition and removes the AllowAll base policy, uninstalls the Windows SDK including `SignTool.exe`, and deletes the entire BAC folder and its contents — fully decommissioning Windows Defender Application Control (WDAC) enforcement on the endpoint. A system reboot is required for changes to take effect.

Important: This task is deprecated and will be removed from BigFix Application Control v2.x in near future.

The task performs the following actions:

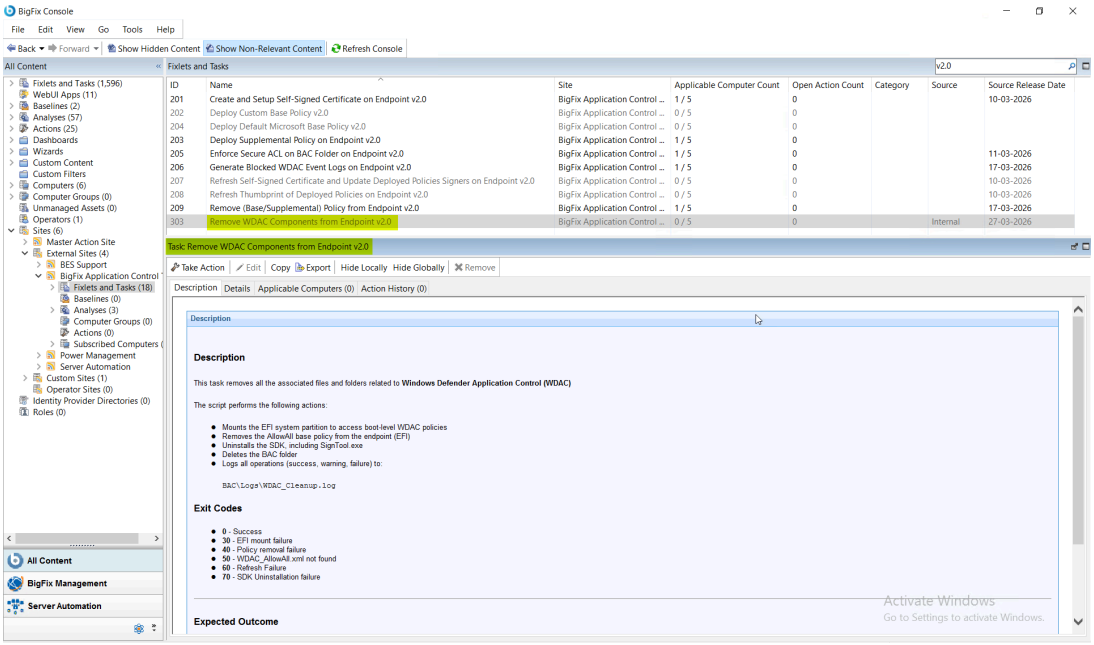
- Mounts the EFI system partition to access the boot-level WDAC policies.
- Removes the AllowAll base policy from the endpoint (EFI).
- Uninstalls the SDK, including SignTool.exe.
- Deletes the BAC folder from the endpoint.
- Logs all operations (success, warning, failure) to the `BAC\Logs\WDAC_Cleanup.log` file.

Refer to the table below to know more about the task's exit code.

Table 48. Remove WDAC Components v2.0: Task Exit Codes and Meanings

Exit Code	Meaning
0	Success
30	EFI mount failure
40	Policy removal failure
50	WDAC_AllowAll.xml file not found.
60	Refresh failure
70	SDK un-installation failure

Figure 53. BigFix Console — Remove WDAC Components from Endpoint v2.0 Task Overview



1. In the BigFix Console, navigate to **All Content > BigFix Application Control > Fixlets and Tasks**.
2. From the **Fixlets and Tasks** pane, select **Task: Remove WDAC Components from Endpoint v2.0**.
3. From the **Task: Remove WDAC Components from Endpoint v2.0** pane, click the **Applicable Computers(n)** tab and view the endpoints on which you want to run the task.
4. Select the **Take Actions** tab and select the endpoints on which you want to apply this installer task.
5. Click **OK**.

A successful execution of this task results in the following outcomes:

- Removes all the associated Application Control files and folders from the endpoint.
- A system reboot is required for the changes to take effect.

Current WDAC Uninstallation Tasks

Uninstall WDAC Components from Target Endpoint *(on page 75)*

Uninstall BAC Configuration from the WDAC Management Node *(on page 73)*

Uninstall BAC Configuration from Root Server *(on page 70)*

Deprecated Tasks: Parent Topic and Policy Removal

Deprecated BigFix Application Control v2.x Fixlets and Tasks *(on page 90)*

Remove Base or Supplemental WDAC Policy from Endpoint v2.0 *(on page 126)*

Related Deprecated Setup Tasks — What This Task Removes

Install and Configure the BigFix Application Control (BAC) Component *(on page 37)*

Create and Set Up a Self-Signed Code-Signing Certificate on Endpoint v2.0 *(on page 98)*

Reference

BigFix Application Control v2.0 Tasks: Exit Codes *(on page 134)*

BigFix Application Control: Troubleshooting WDAC Enforcement Issues *(on page 156)*

View Application Control Endpoint Details in BigFix Web Reports v2.x

This deprecated task describes how to use BigFix Web Reports to view a read-only overview of all Application Control managed endpoints. The report provides three tabs: the **Managed Devices** tab lists all managed endpoints with property and OS filtering and CSV export; the **Policy Details** tab shows the WDAC rules assigned per endpoint; and the **Event Details** tab displays all events executed on each endpoint.

Important: This task is deprecated and will be removed from BigFix Application Control v2.x in near future.

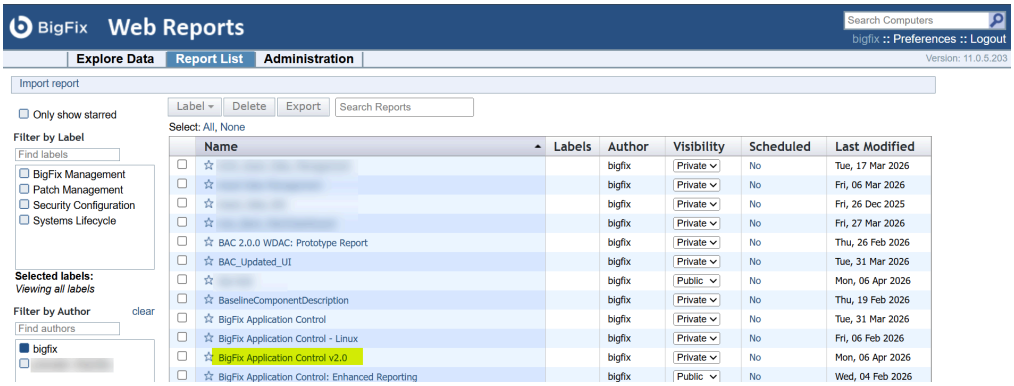
Learn how to use BigFix Web Reports to view a read-only overview of all the Application Control managed endpoints.

As an administrator, you can use the BigFix Web Reports to see a holistic, read-only view of all the endpoints of your enterprise which have BigFix Application Control installed on them.

Follow the steps below to view details from BigFix Web Reports:

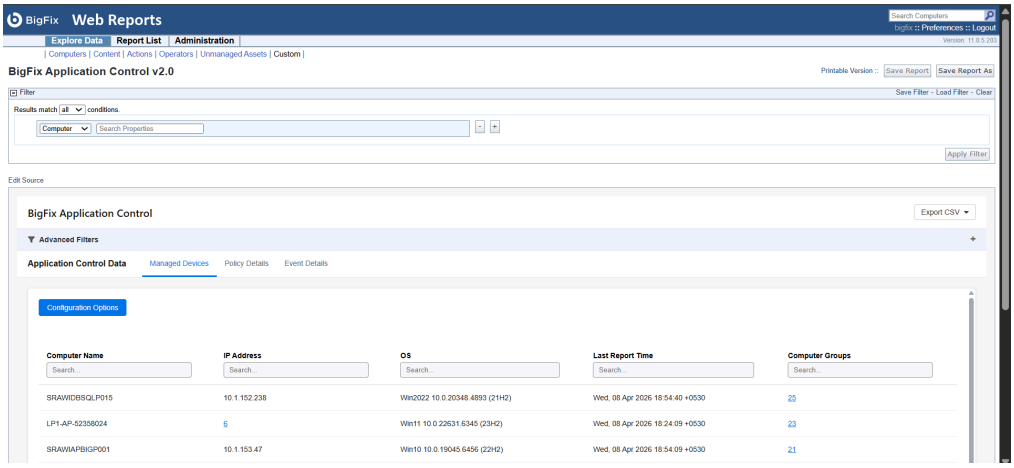
1. Login to **BigFix Web Reports**.
2. On the **Web Reports** home page, select the **Report List** tab and click **BigFix Application Control v2.0**.

Figure 54. BigFix Web Reports — Report List Tab Showing BigFix Application Control v2.0 Report Entry



3. On the BigFix Application Control pane, you will see the following 3 tabs:

Figure 55. BigFix Web Reports — BigFix Application Control v2.0 Report Showing Three Tabs: Managed Devices, Policy Details, and Event Details



a. **Managed Devices**

All the managed endpoints will be listed in this tab in a tabular format. You can filter the managed devices lists using endpoint/BigFix properties. There are two features on this tab: **Configuration Options** & **Export CSV**.

- **Configuration Options**

  This feature lets you add properties or settings that you can use to filter the list of managed devices. We can broadly divide this feature into 3 parts:

  - The first row has a Search field, an OS Filter & Group Filter to filter the list of managed devices.
  - Next rows have the **Add Property** and the **Add Setting** fields. Start typing in the fields to get a list of properties or settings and click **Add Property** or **Add Setting** button as applicable.
  - The last row has the **Rows per page** drop-down where you can set the number of managed devices that are displayed on a page.

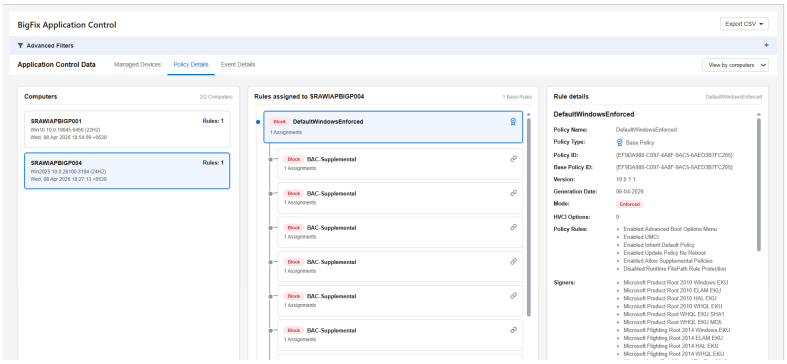
- **Export CSV**

  This feature will export the list of managed devices in CSV format to your machine.

b. **Policy Details**

This tab will display the list of managed devices with rules assigned to them. Select an endpoint or device name from the **Computers** column and the **Rules assigned to <device_name>** column will display all the rules assigned to the endpoint. Selecting a specific rule from it will populate the **Rule details** column.

Figure 56. BigFix Web Reports — Policy Details Tab Showing WDAC Rules Assigned to a Selected Endpoint

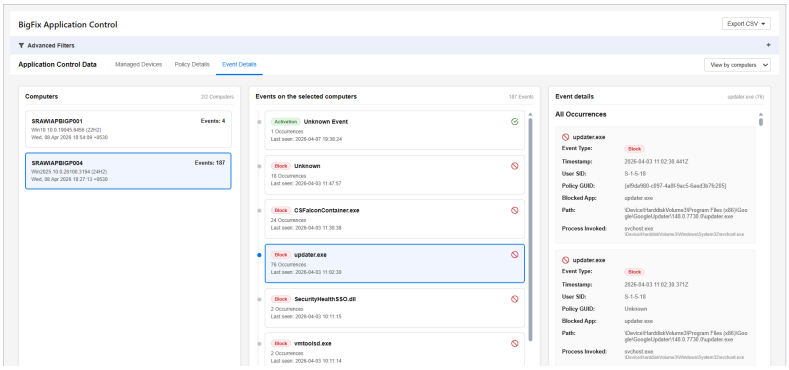


c. **Event Details**

This tab will display the list managed devices with all the events executed on them. Select an endpoint or device name from the **Computers** column and the **Events on the**

selected computers column will list all the events of the endpoint. Selecting a specific event from it will populate the **Event details** column.

Figure 57. BigFix Web Reports — Event Details Tab Showing WDAC Events Executed on a Selected Endpoint



WDAC Reporting and Policy Review

Submit a WDAC Policy for Review *(on page 79)*

Generate Blocked WDAC Event Logs on Endpoint v2.0 *(on page 123)*

Deprecated Tasks: Parent Topic

Deprecated BigFix Application Control v2.x Fixlets and Tasks *(on page 90)*

BigFix Application Control v2.0 Tasks: Exit Codes *(on page 134)*

Target Endpoint Setup and Policy Deployment

Set Up and Initialize WDAC on Target Endpoint *(on page 64)*

Deploy and Refresh WDAC Policy on Target Endpoint *(on page 66)*

Administration and Reference

Administrating BigFix Application Control v2.1.0 *(on page 35)*

BigFix Application Control: Troubleshooting WDAC Enforcement Issues *(on page 156)*

BigFix Application Control v2.0.0 Deprecated Tasks: Exit Code Reference (Tasks 201–209)

This reference topic lists the exit codes for all nine deprecated BigFix Application Control v2.0.0 tasks (201–209). Exit code 0 indicates success; non-zero codes pinpoint the specific failure stage — directory creation, certificate generation, policy processing, binary conversion, signing, EFI deployment, or policy refresh — to enable rapid diagnosis without inspecting log files.

Important: This topic refers to Fixlets or tasks that are deprecated and will be removed from BigFix Application Control v2.0.0 in near future.

Troubleshooting reference for tasks 201–209 with standardized exit codes. Exit code 0 indicates success; non-zero codes pinpoint specific failure stages to aid rapid diagnosis.

Task 201-Create and Setup Self-Signed Certificate on Endpoint v2.0.0

Table 49. Task 201: Create and Setup Self-Signed Certificate on Endpoint v2.0.0 — Exit Code Reference

Exit Code	Description
0	Success.
10	BAC directory creation failure.
11	Certificate directory creation failure.
12	Log directory creation failure.
20	Certificate creation failure.
30	Certificate export failure.

Task 202-Deploy Custom Base Policy v2.0.0

Table 50. Task 202: Deploy Custom Base Policy v2.0.0 — Exit Code Reference

Exit Code	Description
0	Success. The custom WDAC base policy was successfully processed, signed, and deployed.
10	Working Directory Creation Failed. The BAC working directory could not be created or accessed.
11	Log Directory Creation Failed. The logging directory could not be created, preventing execution tracking.
12	Policy Directory Creation Failed. The policy storage directory could not be initialized.
30	Policy Processing Failure. Error occurred during XML processing, version bumping, or rule injection.
40	Binary Conversion Failure.

Table 50. Task 202: Deploy Custom Base Policy v2.0.0 — Exit Code Reference (continued)

Exit Code	Description
	Failed to convert the WDAC policy from XML format to binary (.cip).
50	Signing Failure. The policy signing process failed or generated an invalid signature.
60	Deployment Failure. Failed to mount the EFI partition or copy the policy to the target location.
70	Policy Refresh Failure. The policy refresh process or EFI dismount failed.

Task 203-Deploy Supplemental Policy on Endpoint v2.0.0

Table 51. Task 203: Deploy Supplemental Policy on Endpoint v2.0.0 — Exit Code Reference

Exit Code	Description
0	Success.
20	Missing Base Policy GUID.
30	No valid rules generated.
40	Signing failure or invalid signed output.

Task 204-Deploy Default Microsoft Base Policy v2.0.0

Table 52. Task 204: Deploy Default Microsoft Base Policy v2.0.0 — Exit Code Reference

Exit Code	Description
0	Success. The WDAC base policy was successfully processed, signed, and deployed.
10	Working Directory Creation Failed. The BAC working directory could not be created or accessed.
11	Log Directory Creation Failed. The logging directory could not be created, preventing execution tracking.
12	Policy Directory Creation Failed. The policy storage directory could not be initialized.
20	Template Missing.

Table 52. Task 204: Deploy Default Microsoft Base Policy v2.0.0 — Exit Code Reference (continued)

Exit Code	Description
	The selected WDAC policy template file was not found or failed to download.
30	Policy Processing Failure. An error occurred during policy preparation, versioning, or rule configuration.
40	Binary Conversion Failure. Failed to convert the WDAC policy from XML format to binary (.cip).
50	Signing Failure. The policy signing process failed or the signed output was not generated correctly.
60	Deployment Failure. Failed to mount the EFI partition or copy the policy to the target location.
70	Policy Refresh Failure. The policy refresh process failed after deployment. A reboot may be required.

Task 205-Enforce Secure ACL on BAC Folder on Endpoint v2.0.0

Table 53. Task 205: Enforce Secure ACL on BAC Folder on Endpoint v2.0.0 — Exit Code Reference

Exit Code	Description
0	Success.
10	BAC directory not found.
20	Failed to apply ACL permissions.
30	Failed to set ownership.

Task 206-Generate Blocked WDAC Event Logs on Endpoint v2.0.0

Table 54. Task 206: Generate Blocked WDAC Event Logs on Endpoint v2.0.0 — Exit Code Reference

Exit Code	Description
0	Success.
10	Invalid time frame.

Table 54. Task 206: Generate Blocked WDAC Event Logs on Endpoint v2.0.0 — Exit Code Reference (continued)

Exit Code	Description
20	BAC path not found.
30	Event query failure.
40	No events found.
50	Report generation failure.

Task 207-Refresh Self-Signed Certificate and Update Deployed Policies Signers on Endpoint v2.0.0

Table 55. Task 207: Refresh Self-Signed Certificate and Update Deployed Policies Signers on Endpoint v2.0.0 — Exit Code Reference

Exit Code	Description
0	Success.
10	Directory setup failure.
20	Certificate creation or export failure.
30	XML processing or version bump failure.
40	Binary conversion failure.
50	Sign tool signing failure.
60	EFI mount or deployment failure.
70	Policy refresh tool failure.

Task 208-Refresh Thumbprint of Deployed Policies on Endpoint v2.0.0

Table 56. Task 208: Refresh Thumbprint of Deployed Policies on Endpoint v2.0.0 — Exit Code Reference

Exit Code	Description
0	Success.
10	Directory setup failure.
20	Certificate creation or export failure.
30	XML processing or version bump failure.
40	Binary conversion failure.
50	Sign tool signing failure.

Table 56. Task 208: Refresh Thumbprint of Deployed Policies on Endpoint v2.0.0 — Exit Code Reference (continued)

Exit Code	Description
60	EFI mount or deployment failure.
70	Policy refresh tool failure.
80	Thumbprint rotation failure.

Task 209-Remove (Base/Supplemental) Policy from Endpoint v2.0.0

Table 57. Task 209: Remove Base or Supplemental Policy from Endpoint v2.0.0 — Exit Code Reference

Exit Code	Description
0	Success.
10	Invalid input parameters.
20	BAC path not found.
30	EFI mount failure.
40	Policy removal failure.
50	AllowAll.xml not found.
60	AllowAll deployment failure.

Deprecated Certificate and Policy Deployment Tasks (201–204)

Task 201: Create and Set Up a Self-Signed Code-Signing Certificate on Endpoint v2.x *(on page 98)*

Task 202: Deploy Custom Base Policy v2.x *(on page 109)*

Task 203: Deploy Supplemental Policy on Endpoint v2.x *(on page 113)*

Task 204: Deploy Default Microsoft Base Policy v2.x *(on page 101)*

Deprecated ACL, Event Logging, and Certificate Refresh Tasks (205–208)

Task 205: Enforce Secure ACL Permissions on the BAC Folder on Endpoint v2.x *(on page 121)*

Task 206: Generate Blocked WDAC Event Logs on Endpoint v2.x *(on page 123)*

Task 207: Refresh the Self-Signed Certificate and Update Deployed WDAC Policies Signers on Endpoint v2.x *(on page 116)*

Task 208: Refresh the Thumbprint of Deployed WDAC Policies Signers on Endpoint v2.x *(on page 119)*

Deprecated Policy Removal Task and Parent Topic (209)

Task 209: Remove Base or Supplemental WDAC Policy from Endpoint v2.x *(on page 126)*

Deprecated BigFix Application Control v2.x Fixlets and Tasks *(on page 90)*

Troubleshooting

BigFix Application Control: Troubleshooting WDAC Enforcement Issues *(on page 156)*

Chapter 12. BigFix Application Control v2.x: Frequently Asked Questions

This FAQ covers BigFix Application Control v2.x across six topic areas: product fundamentals and architecture, Base and Supplemental Policy design, WDAC policy signing and TPM key protection, Secure Boot and boot recovery, phased deployment and enforcement introduction, Code Integrity telemetry, rule level selection (Publisher, Hash, Path, and File Name), policy naming conventions, and deprecated v2.x distributed-model scenarios.

Application Control v2.x FAQs

What is BigFix Application Control 2.0?

BigFix Application Control 2.0 provides centralized management and deployment of Windows Defender Application Control (WDAC) policies through the BigFix platform. It enables administrators to create, validate, sign, publish, deploy, and monitor WDAC policies while Windows Code Integrity performs the actual application authorization and enforcement on Windows endpoints.

What does BigFix Application Control 2.0 control?

Application Control manages which applications are permitted or prevented from executing on managed Windows endpoints based on the configured WDAC policy. Policies can use supported WDAC rules such as:

- Publisher
- File hash
- File path
- Signer/certificate-based rules

The exact enforcement behavior depends on the rules and options configured in the Base and Supplemental Policies.

What is the difference between BigFix and Windows Code Integrity?

BigFix provides the management, deployment, and monitoring layer. Windows Code Integrity / WDAC provides the local enforcement layer. In other words: BigFix manages and deploys the policy → Windows Code Integrity enforces the policy.

What is a Base Policy?

The Base Policy establishes the foundational WDAC security posture for an endpoint. It defines the primary trust model, policy configuration, and core application authorization rules on which Supplemental Policies depend.

What is a Supplemental Policy?

A Supplemental Policy extends an existing Base Policy with additional application authorization rules. A Supplemental Policy does not replace the Base Policy.

Can multiple Supplemental Policies be deployed?

Yes. Multiple Supplemental Policies can extend a Base Policy, subject to the Windows WDAC policy architecture and applicable policy relationships.

Does deploying or updating a policy require a system restart?

The solution supports policy updates without a reboot where supported by the Windows platform and configured policy options. However, reboot requirements can vary depending on the Windows version/build, policy type, HVCI/Memory Integrity configuration, and policy-signing configuration.

How are WDAC policies protected from unauthorized modification?

Policies requiring signed updates are digitally signed before deployment. Where TPM-backed signing is configured, the private signing key is protected by the TPM and is not exported during the normal signing workflow.

Is the signing private key stored on the BigFix server?

No. Policy signing is performed within the designated WDAC Management environment. The BigFix publication and deployment infrastructure receives the signed policy artifact and does not require access to the private signing key.

What happens if the primary signing certificate becomes unavailable?

An emergency/recovery signing capability can be maintained as part of the policy trust configuration. The emergency signer can support approved recovery scenarios such as primary signing infrastructure failure, certificate expiration or rotation, or loss of the primary signing capability.

Should the emergency signer be configured in advance?

Yes. The emergency/recovery signer should be configured in advance as part of the Base Policy trust configuration. Preconfiguring the signer allows an authorized recovery signer to be trusted if the primary signing infrastructure becomes unavailable or a recovery operation is required.

Can the signing certificate be rotated?

Yes. Certificate rotation is supported as part of the signing certificate lifecycle. The replacement certificate must be incorporated into the appropriate policy trust configuration before it is required for policy updates.

Does the solution require a TPM?

TPM/vTPM is recommended for environments requiring hardware-backed protection of policy-signing keys. Where TPM-backed signing is configured, the signing private key is protected by the TPM on the designated management system.

Which Windows versions are currently supported?

The current architecture targets:

- Windows 10 version 1903 and later
- Windows 11
- Windows Server 2022 and later

Specific WDAC capabilities and policy behavior can vary by Windows release and build.

Does Application Control require Secure Boot?

Secure Boot is an important part of the trust chain for signed WDAC policies, particularly for early-boot policy validation. Customers should maintain the supported Secure Boot configuration for production systems using signed Application Control policies.

What happens if a WDAC policy prevents Windows from starting?

A policy that blocks a boot-critical component can potentially result in a boot failure. Recovery may require Windows recovery mechanisms, policy replacement or removal, firmware-level recovery procedures, or manual intervention.

If the endpoint uses BitLocker, a boot or Secure Boot configuration change may also trigger BitLocker recovery. Customers should ensure that their established BitLocker recovery process is available for applicable recovery scenarios.

Does Boot Audit on Failure guarantee recovery?

No.

Boot Audit on Failure is a mitigation for applicable boot-critical driver blocking scenarios. It does not guarantee recovery from every type of boot failure or protect against all forms of policy tampering.

What happens if Secure Boot is disabled?

Disabling or changing Secure Boot configuration can affect the trusted boot state and may trigger BitLocker recovery on systems where BitLocker protection is enabled.

Customers using BitLocker should ensure that their established BitLocker recovery process is available before changing Secure Boot settings during testing or recovery.

Can a local administrator remove or modify a deployed policy?

The architecture does not provide complete protection against every form of local administrative tampering.

Organizations should therefore use appropriate operating-system administrative controls in addition to Application Control.

What happens if the BigFix Agent itself is blocked?

If the BigFix Agent is blocked by the Application Control policy, the endpoint may not be able to receive remediation actions through BigFix.

Manual recovery may therefore be required in such scenarios.

Which Code Integrity events can be used to track Application Control policy and enforcement status?

Windows provides multiple Code Integrity events in the *Microsoft-Windows-CodeIntegrity/Operational* log. The following events are particularly useful for Application Control troubleshooting and enforcement verification:

Table 58. Event - Use Table

Event	Use
3076	Main App Control block event in Audit Mode. Indicates that the file would have been blocked if the policy were enforced.
3077	Main App Control block event in Enforcement Mode. Indicates that the file was blocked by the policy.
3089	Provides signature information associated with a blocked or audit-blocked file and can help investigate the reason for the policy decision.
3099	Indicates that an App Control policy was successfully loaded.

Additional Code Integrity events may be generated depending on the Windows version, policy configuration, and scenario.

How can the generated WDAC policy XML be reviewed before deployment?

The generated policy XML can be reviewed using the Submit a WDAC XML Policy for Review *(on page 79)* task provided by BigFix Application Control.

This task allows administrators to review the generated policy XML before publishing or deploying the policy and verify that the policy configuration and intended rules are correct.

How does BigFix determine whether a policy was successfully deployed?

BigFix deployment status can be used to verify whether the policy deployment action completed successfully on the endpoint.

The Windows Code Integrity event log should be used separately to verify policy loading and application enforcement behavior.

Does BigFix continuously collect Code Integrity events?

No. The current design uses an on-demand collection model. Telemetry is collected through explicit BigFix actions rather than continuously streaming all Code Integrity events from every endpoint.

How much historical Code Integrity data is collected?

The current reporting design limits collection to a maximum 14-day lookback window.

Does BigFix itself enforce application execution?

No.

BigFix manages, deploys, and monitors the policy. Windows Code Integrity performs the actual application authorization and enforcement.

Can policies be updated without modifying the Base Policy?

Yes, where the requirement can be addressed through a Supplemental Policy.

Supplemental Policies provide a mechanism for adding application authorization rules while keeping the foundational Base Policy unchanged.

What happens if a Supplemental Policy fails?

A Supplemental Policy does not replace the Base Policy. The Base Policy remains the foundational policy and continues to establish the endpoint's underlying security posture.

Is the Intelligent Security Graph (ISG) included?

ISG is not included in the current release.

Is Managed Installer supported?

Managed Installer is not included in the current release.

Is kernel/HVCI enforcement included?

No. The current release focuses on user-mode application control.

Deeper kernel-level scenarios, including comprehensive WHQL and HVCI validation, are outside the current validation scope.

Does the solution support Microsoft Active Directory Certificate Services (AD CS)?

Microsoft AD CS / Enterprise PKI integration is not available in the current release.

Can customers use their own certificate authority?

Customer-managed certificate infrastructure can be used where it meets the supported WDAC policy-signing requirements.

Is AppID Tagging supported?

AppID Tagging is not included in the current release.

How are policies distributed to endpoints?

The general deployment flow is:

Policy Creation → Validation → Review → Signing → BigFix Publication → Endpoint Deployment → Windows Code Integrity Activation → Enforcement and Reporting.

What happens when a policy is retired?

The active policy should not simply be deleted from an endpoint. The supported retirement workflow establishes an appropriate replacement or unblocking state before the retired policy is removed.

What should customers validate before enabling enforcement?

Customers should validate:

- Business-critical applications
- BigFix Agent functionality
- Required network drivers and services
- Boot-critical components
- Security/EDR software
- Application installers and updaters
- Base and Supplemental Policy behavior
- Generated policy XML using the Review Fixlet
- Recovery procedures
- Signing and recovery-signer configuration

Should Application Control be deployed to the entire environment at once?

No. A phased deployment is recommended:

Test/Audit → Review → Refine → Validate → Pilot Enforcement → Broader Enforcement

Microsoft recommends deploying App Control policy changes in Audit Mode before proceeding to Enforcement and using deployment rings to control the rollout.

What are the main limitations of the current release?

The current release has the following scope limitations:

- ISG is not included.
- Managed Installer is not included.
- Advanced kernel/HVCI validation is outside the current validation scope.
- AppID Tagging is not included.
- AD CS / Enterprise PKI integration is not available.
- Certain boot failures may require manual or offline recovery.
- Complete protection against local administrative tampering is not provided.
- Blocking the BigFix Agent may require manual recovery.
- Code Integrity telemetry uses an on-demand collection model with a 14-day lookback limit.
- Some WDAC behaviors vary by Windows version and build.

Where can administrators check why an application was blocked?

Administrators can review the *Microsoft-Windows-CodeIntegrity/Operational* event log for events related to the blocked application.

The applicable Base and Supplemental Policy configuration can also be reviewed to determine whether an explicit deny rule applies to the application.

How should Application Control enforcement be introduced?

Application Control enforcement should be introduced in a phased approach:

1. Deploy the policy in Audit Mode to a test environment.
2. Exercise normal application usage and review the Code Integrity events generated by applications that would be blocked under enforcement.
3. Review and refine the policy based on the observed results.
4. Review the generated policy XML using the Review Fixlet.
5. Validate the updated policy in the test environment.
6. Deploy the policy in Enforcement Mode to a controlled pilot group.
7. Monitor the pilot environment and confirm expected application behavior.
8. Gradually expand deployment to additional production groups.

Recommended approach:

- **Audit → Review → Refine → Validate → Pilot Enforcement → Broader Enforcement**

Microsoft recommends deploying App Control policy changes in Audit Mode before proceeding to Enforcement, monitoring the resulting events, and using deployment rings for controlled rollout.

What is the recommended recovery strategy?

Customers should establish recovery procedures before enabling enforcement. The recovery plan should account for:

- An authorized emergency signing capability.
- Secure Boot recovery procedures.
- Policy rollback/replacement procedures.
- Offline/manual recovery capability.
- Controlled pilot deployment.
- Signing certificate lifecycle management.
- For endpoints using BitLocker, the organization's established BitLocker recovery process should be available for applicable recovery scenarios.

What is the security boundary in the architecture?

The architecture separates:

- Policy Management → Policy creation and validation
- Policy Signing → Cryptographic trust and signing-key protection
- BigFix Publication → Controlled policy distribution
- BigFix Deployment → Endpoint targeting and deployment
- Windows Code Integrity → Local policy validation and enforcement

This separation limits the need for deployment infrastructure to have access to policy-signing private keys.

What is the recommended deployment model?

The recommended approach is:

Establish signing and recovery trust → Create and validate the Base Policy → Review the generated XML → Test applications and recovery scenarios in Audit Mode → Refine and validate → Deploy to a pilot group in Enforcement Mode → Monitor → Expand deployment gradually.

Is there a naming convention for policies?

Yes. Policy names must follow the required naming convention:

- Base Policy: The name must start with Base_.
- Supplemental Policy: The name must start with Supp_.
- The name may contain only alphanumeric characters (A–Z, a–z, 0–9) and underscores (_).
- Spaces and special characters are not permitted, including -, ., /, \, @, #, %, &, (,), and :.

Table 59. Policy Type Example Names

Policy Type	Valid Example	Invalid Example
Base	Base_StandardPolicy	Base-StandardPolicy
Supplemental	Supp_ChromePolicy	Supp Chrome Policy
Base	Base_Windows11_01	Base.Windows11.01
Supplemental	Supp_Chrome_01	Supp@Chrome#01

When can the Package rule type be used for an application?

The **Package** rule type can be used for an application installed as an **AppX/MSIX packaged application**, typically under `C:\Program Files\WindowsApps`.

Verify it using below PowerShell command:`Get-AppxPackage *<application-name>* | Select Name, PackageFamilyName, InstallLocation`

If the application has a **Package Family Name (PFN)** and is installed as an AppX/MSIX package, it can be considered for a **Package** rule.

When should I use the COM rule type?

COM (Component Object Model) is a Windows technology that allows applications to use functionality provided by other software components. Use the **COM** rule type when a specific COM object (identified by its **CLSID**) needs to be **allowed or explicitly denied** by the App Control policy. This can be useful when an application requires a COM object that is not allowed by the policy, or when a specific COM object needs to be blocked.

The CLSID can be identified from **Event ID 8036 (COM object blocked)** in Event Viewer or from the Windows Registry.

What should I do if the application binary cannot be installed, downloaded, or made available on the designated WDAC Management Node?

If the application binary cannot be made available on the designated WDAC Management Node, the required application rule can be created manually using the **Microsoft App Control Policy Wizard** as an alternative workflow.

1. Open the App Control Policy Wizard: App Control Policy Wizard
2. Create the required application rule using the appropriate rule type and application metadata.
3. Generate the WDAC policy XML from the wizard.
4. Copy the generated XML content and provide it to the Custom Policy XML Generation workflow in BigFix Application Control v2.x.
5. Continue with the standard BigFix Application Control workflow for policy signing, publication, deployment, and enforcement through the applicable Fixlet/Task.

Deprecated Application Control v2.x FAQs

Important:

- Tasks from Application Control v2.x that used the distributed model — including per-endpoint self-signed certificate creation, direct Base and Supplemental Policy deployment on individual endpoints, and related Fixlets — are deprecated.
- Refer to the Deprecated Application Control v2.x Fixlets and Tasks *(on page 90)* chapter for reference documentation on those tasks.
- The FAQs in this section are applicable to the deprecated Application Control v2.x only.

Customer scenario: In Application Control 2.0, a customer needs to block a single application. How can this use case be implemented?

In Application Control 2.0, first, the base policy needs to be deployed, which is a Default Deny (also known as Zero Trust) policy. Next, you can deploy supplementary policies to allow specific applications. This may appear to need more steps, but it is more secure, as it does not allow the application process to start by itself. For convenience, though, Application Control 1.0 is available to customers who would like to continue using that version.

- Step 1: Base Policy - Deny All (except for necessary Microsoft processes, BigFix agent, etc.)
- Step 2: Supplementary Policy - Allow (Application)

So, in Application Control 2.0, only the approved software runs; everything else is blocked by default. The working mechanism of Application Control 2.0 is based on Windows Defender Application Control (WDAC), which has this limitation. In WDAC, you first deploy the policy in audit mode, then check which applications run in your environment, and add those apps to your allowlist. Once you have confirmed the allowlisted applications, you deploy the policy in enforced mode.

So, it is *not possible* to block a single application *directly*. You have to *follow the above 2-step process to implement this use case*. Hence, in this scenario, first deploy the Default Deny base policy, then deploy the supplementary policy allowing all the applications you need. Do not include the application you want to block in the supplementary policy, and you will fulfill the scenario above.

Customer scenario: Supplementary policy action fails if only the application name is specified in Application Control 2.0.

Problem statement: A customer needs to apply the following 2 scenarios using only the name of the application in Application Control 2.0:

- Apply Default Microsoft™ Base Enforced Policy and then allow only specific applications for users.
- Apply Default Microsoft™ Base AllowAll Policy and then block only specific applications for users.

The Supplementary policy has to be applied using the application name so as not to depend on the application's installation path. The supplementary policy is successfully applied using the file path name, but when specifying the application name (such as chrome.exe), the action fails with exit code 30. In the supplementary policy log file, there is the warning `[WARN] Path not found: chrome.exe`.

Note: The customer was using Application Control 1.0 with no issues.

Additional question: Since Application Control 2.0 executes PowerShell scripts, what happens if the PowerShell application is blocked by a policy?

Solution: Refer to the following points for a solution to this scenario:

- Application Control 2.0 does not support using only application names when creating supplementary policies; one has to provide the entire path of the executable/application to create the supplementary policies.
- Application Control 2.0 relies entirely on PowerShell scripts. PowerShell execution should be allowed for deployment, management, and configuration of Application Control 2.0 on the endpoints. The answer to this additional question is to allow PowerShell execution from the BESClient folder or, more specifically, from the site folder under __BESData.
- Lastly, Supplemental Policies only work to extend the allowlist of the base policy. They do not facilitate extending the block list.

Customer scenario: Deployment failures of the "Deploy Default Microsoft Base Policy v2.0" task on a Windows 11 machine

Problem statement: Deployment failures of the "Deploy Default Microsoft Base Policy v2.0" task on a Windows 11 machine were caused by existing WDAC policies and security enforcement mode. The machine was already in Enforcement Mode with HVCI active, which likely conflicted with the

deployment. The issue was identified as being caused by existing WDAC policies on the endpoint that conflicted with the deployment.

Solution: This issue is caused by existing WDAC policies on endpoints that conflict with the deployment. Application Control 2.0 will introduce a sanitization step as a prerequisite in the initial deployment phase to ensure that any existing WDAC policies on the machine do not interfere with the new deployment.



Note: This scenario is still under technical review. When a solution is provided, the solution in this FAQ will be updated.

Common Allowlisting Use Cases for Rule Levels in Deprecated WDAC / Deprecated Application Control 2.0

Publisher Rule Level

- **Recommended Use Case**

  Publisher-based rules are the preferred and most scalable approach when:

  - The application binaries are digitally signed.
  - The certificate chain is trusted by the Windows kernel.
  - Future application updates are expected.

- **Advantages**
  - Automatically trusts newer versions signed by the same publisher.
  - Reduces operational overhead.
  - Avoids frequent policy updates after application upgrades.
- **Example**

  Applications such as: Microsoft Office, Google Chrome, and Adobe Acrobat can usually be trusted using publisher rules because their signing certificates are recognized and trusted by the operating system.

- **Limitations**

  Publisher rules may fail when:

  - The application is unsigned.
  - The certificate chain is incomplete or untrusted.
  - Kernel-mode trust validation rejects the signer.

- **Typical Scenarios**

  Some applications may require multiple supporting binaries and DLLs that are blocked under restrictive WDAC base policies such as `DefaultWindows_Enforced`. In such scenarios, Publisher rules may fail because the application certificate chain is not fully trusted by the Windows kernel during policy evaluation. For example:

  - firefox.exe
  - notepad++.exe

Hash Rule Level

- **Recommended Use Case**

  Hash-based rules should be used when:

  - Publisher rules are not working.
  - The application signer is not trusted.
  - Specific binaries must be tightly controlled.
  - Security requirements demand exact file integrity validation.
- **Advantages**
  - Most secure rule type.
  - Trusts only the exact approved binary.
  - Prevents unauthorized file replacement.
- **Example**

  Applications with multiple dependent binaries and DLLs may require Hash-based allowlisting to ensure all required components are individually trusted. For example, the installation path: `C:\Program Files\Mozilla Firefox\*`
- **Limitations**
  - Any file modification or upgrade changes the hash.
  - Policies must be updated after every application update.
- **Typical Scenarios**

  Applications containing the following often require Hash-based allowlisting:

  - Multiple DLL dependencies.
  - Embedded runtime components.
  - Non-standard signing chains.

Path Rule Level

- **Recommended Use Case**

  Path-based rules are useful when:

  - Applications are installed in controlled directories.
  - Frequent updates occur.
  - Managing individual hashes is operationally difficult.
- **Advantages**
  - Simplifies management.
  - Automatically allows updated binaries within the approved directory.
  - Suitable for enterprise-managed application folders.
- **Example**

  A Path rule can be used when the installation directory is protected and writable only by administrators. For example, the path: `C:\Program Files\Mozilla Firefox\*`.

- **Limitations**
  - Less secure compared to Hash or Publisher rules.
  - Any executable copied into the approved path may also run.
  - Requires strict NTFS permission controls.
- **Typical Scenarios**

  Commonly used for:

  - Enterprise application installations.
  - Vendor-managed software directories.
  - Internal packaged applications.

File Name Rule Level

- **Recommended Use Case**

  File Name rules are suitable when:

  - Specific executable names must be allowed regardless of location.
  - Legacy applications have inconsistent installation paths.
  - Operational simplicity is prioritized.
- **Advantages**
  - Simple implementation.
  - Useful for legacy environments.
- **Example**

  Allowing executables solely based on file name may unintentionally trust malicious files renamed to approved names. For example:

  - firefox.exe
  - notepad++.exe
- **Limitations**
  - Weak security model.
  - Malicious binaries can be renamed to approved file names.
  - Not recommended for high-security environments.
- **Typical Scenarios**

  Used primarily for:

  - Testing environments.
  - Temporary exceptions.
  - Legacy operational compatibility.

Recommended Enterprise Approach

Table 60. Rule Level Summary

Rule Level	Security Level	Recommended Usage
Publisher	High to Low	Preferred for trusted signed applications.
Hash	Very High to High	Best for untrusted or sensitive binaries.
Path	Medium to Low	Suitable for controlled installation directories.
File Name	Low to Low	Only for temporary or legacy scenarios.

General Recommendation

When applications require multiple supporting binaries and DLLs that are blocked under restrictive WDAC base policies, and the application signing chain is not fully trusted by the Windows kernel:

- Publisher rules may not be reliable.
- Hash-based allowlisting is generally the most secure approach.
- Path-based rules may be considered if the installation directory is tightly controlled.
- File Name rules should be avoided in production environments wherever possible.

Product Context and Architecture

BigFix Application Control: Product Overview

System Architecture

Policy Design and Behavior

BigFix Application Control v2.x: Policy Behavior and Deployment Considerations *(on page 32)*

BigFix Application Control: WDAC Known Limitations *(on page 18)*

Administrator Task Reference

Administering BigFix Application Control v2.x *(on page 35)*

Provision the TPM-Backed Code-Signing Certificate on the WDAC Management Node *(on page 44)*

Manage WDAC Policies on the WDAC Management Node *(on page 49)*

Submit Policy for Review *(on page 79)*

Set Up and Initialize WDAC on Target Endpoints *(on page 64)*

Deploy and Refresh WDAC Policies on Target Endpoints *(on page 66)*

Migration and Deprecated Tasks

Migrating from BigFix Application Control v1.0.0 to v2.x *(on page 97)*

Deprecated Application Control v2.x Fixlets and Tasks *(on page 90)*

Troubleshooting

Troubleshooting Application Control *(on page 156)*

Chapter 13. BigFix Application Control: Troubleshooting WDAC Enforcement Issues

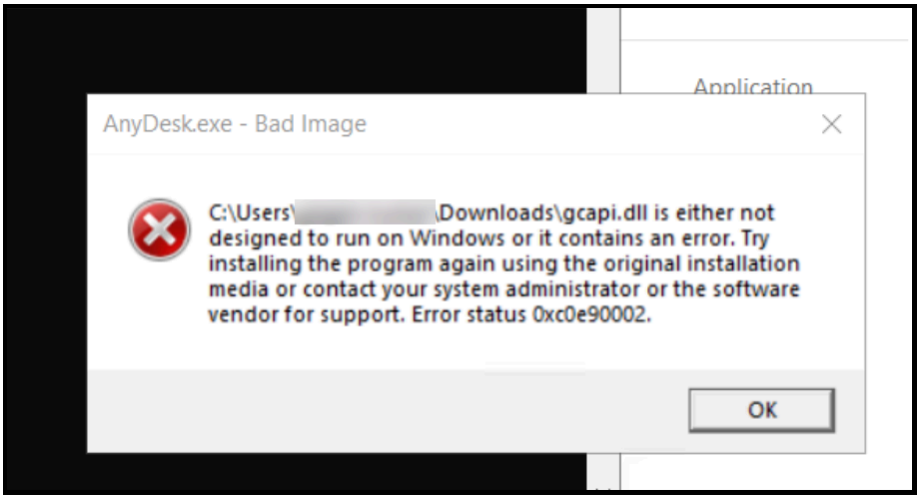
This reference covers three diagnosed WDAC enforcement issues in BigFix Application Control environments: (1) AnyDesk blocked with error 0xc0e90002 (STATUS_CODE_INTEGRITY_VIOLATION) due to an untrusted Google LLC DLL extracted to the temp folder; (2) Postman displaying "Failed to extract installer" because WDAC silently blocks background child processes before the native Device Guard notification appears; and (3) an application blocked by Event ID 3077 despite an explicit allow rule, caused by a FilePublisher or FileAttrib MinimumFileVersion threshold mismatch.

Issue 1: `Anydesk.exe` bad image

Error code: 0xc0e90002 (STATUS_CODE_INTEGRITY_VIOLATION)

Error message: `C:\Users\...\Downloads\gcapi.dll` is either not designed to run on Windows or it contains an error.

Figure 58. AnyDesk.exe STATUS_CODE_INTEGRITY_VIOLATION (0xc0e90002) — gcapi.dll Blocked by WDAC Signer Policy



Root cause: When AnyDesk™ runs portably, it extracts `gcapi.dll` (a Google™ dependency) into your session temp folder (`C:\Users\...\AppData\Local\Temp\5\gcapi.dll`). Your Windows Defender Application Control (WDAC) policy blocks this DLL from loading because `gcapi.dll` is signed by **Google LLC**, whereas your existing AnyDesk™ policy rule only trusts **AnyDesk Software GmbH**. Because code integrity blocks the untrusted DLL from loading into memory, Windows® throws the *0xc0e90002 Bad Image* error.

Solution:

1. **Step 1**:Use **Task ID: 354** to create a base policy including an allow AnyDesk™ rule using publisher providing the **.exe** path through custom parameter rule for the path `C:\Users\Administrator\Downloads\AnyDesk.exe`.
2. **Step 2**: Use **Task ID: 355** to sign, compile, and publish it.
3. **Step 3**: Use **Task ID: 406** to deploy and enforce the policy.
4. **Result**: AnyDesk™ does not show any bad image error.

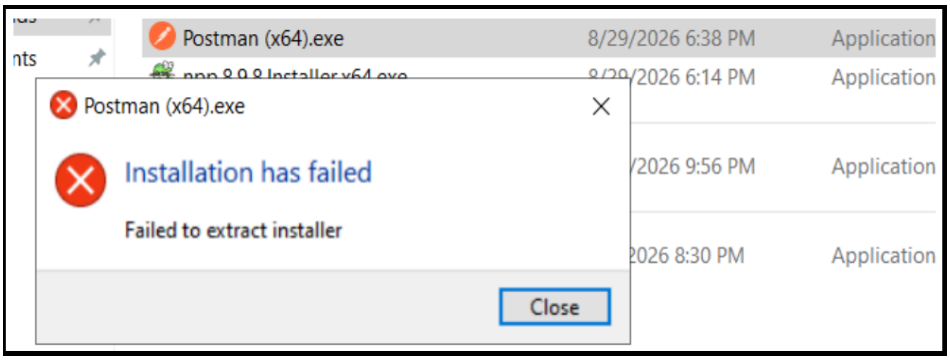
Note: If any older version of AnyDesk is being used then while creating a base policy include an allow rule for **anydesk.exe** as well as the **gcapi.dll** via publisher rule using the **Task ID: 354**. **Step 2** and **Step 3** remains the same.

Issue 2: Failed to extract installer

Error code: Installation has failed: Failed to extract installer

Error message: Postman displays its own UI error *Failed to extract installer* instead of allowing Windows to show its native *blocked by your organization's Device Guard policy* pop-up notification.

Figure 59. Postman "Failed to Extract Installer" — WDAC Block on Background Child Process Suppresses Native Device Guard Notification



Root cause: Postman(x64).exe runs an installer framework that catches process failures internally. When Windows Defender Application Control (WDAC) blocks its temporary extraction files in the background, Postman displays its own UI error *Failed to extract installer* instead of allowing Windows to show its native *blocked by your organization's Device Guard policy* pop-up notification.

This behavior occurs because Postman(x64).exe is an installer wrapper. When executed, it silently unpacks background installer binaries into `%LocalAppData%\Temp\` and launches them as child processes. WDAC intercepts and blocks these background child processes. Because the block happens to a secondary process in the background, the main Postman wrapper handles the process crash internally and displays its own custom application error box before Windows can trigger its native OS block dialog.

Solution: To resolve this issue, eliminate the custom application pop-up, and show the native Windows WDAC block notification to the customer, follow these exact steps:

1. **Step 1**:Use **Task ID: 354** to create a base policy including an allow rule for Postman™ using Publisher rule, providing the **.exe** path through custom parameter rule for the path: `C:\Users\Administrator\Downloads\Postman (x64).exe`.
2. **Step 2**: Use **Task ID: 355** to sign, compile, and publish it.
3. **Step 3**: Use **Task ID: 406** to deploy and enforce the policy.
4. **Result**: The Postman™ installs and launches cleanly with no *Failed to extract installer* error, since the extracted **Postman.exe** is now allow-listed by the publisher.

Issue 3: Binary blocked by minimum version threshold

Error code: An executable was blocked (Event ID 3077) even though an explicit rule to allow the executable exists in the policy XML.

Error message:

```
Code Integrity determined that a process (\Device\HarddiskVolume3\Windows\explorer.exe)
attempted to load \Device\HarddiskVolume3\Program Files\Notepad++\notepad++.exe
that did not meet the Enterprise signing level requirements or violated code integrity policy
(Policy ID:{9c8c2d01-ce0f-40ad-bb53-806ce27d7fb8}).
```

Root cause: Rules created at the **FilePublisher** or **FileAttrib** level enforce a MinimumFileVersion.

```
<FileAttrib ID="ID_FILEATTRIB_NPP" FileName="notepad++.exe" MinimumFileVersion="8.9.6.4" />
```

This block occurs for two main reasons:

1. **Installed Version Below Threshold:** The endpoint runs a version (e.g., 8.9.6.2) lower than the policy requirement (8.9.6.4).
2. **PE Header Mismatch:** WDAC Wizard reads the internal PE header FileVersion (8.9.6.4), but Code Integrity evaluates against the local ProductVersion (8.9.6.2). Since 8.9.6.2 < 8.9.6.4, evaluation fails.



Note: Event 3077 does not log version numbers directly.

Verification: Check the installed file attributes in PowerShell:

```
(Get-Item "C:\Program Files\Notepad++\notepad++.exe").VersionInfo | Select-Object FileVersion,
 ProductVersion
```

Compare the output against **MinimumFileVersion** in `WDAC_BasePolicy.xml`.

Solution:

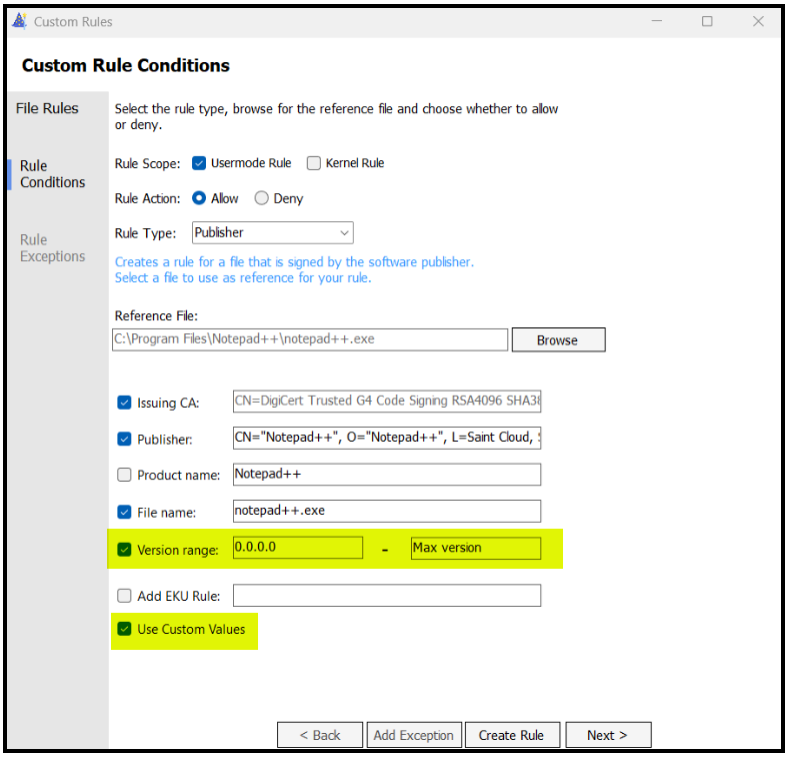
1. **Step 1 (Option 1)**: Use **Task ID: 354** to create a base policy including an allow rule for Notepad ++™ using File attribute and Set **MinimumFileVersion** in `WDAC_BasePolicy.xml` to match or precede the installed version:

```
<FileAttrib ID="ID_FILEATTRIB_NPP" FileName="notepad++.exe"
 MinimumFileVersion="8.9.6.2" />
```

OR

Step 1 (Option 2): Change the Minimum Version in the App Control Policy Wizard and regenerate the XML to match or precede the installed version.

Figure 60. WDAC Policy Wizard Custom Rule Conditions — Setting MinimumFileVersion to Match Installed Endpoint Version



2. **Step 2**: Use **Task ID: 355** to sign, compile, and publish it.
3. **Step 3**: Use **Task ID: 406** to deploy and enforce the policy.
4. **Result**: The Postman™ installs and launches cleanly with no *Failed to extract installer* error, since the extracted **Notepad++.exe** is now allow-listed by the publisher.

Issue 4: MobaXterm® bad image, required unsigned DLL dependencies are blocked by WDAC

Error Code: 0xc0e90002 is displayed

Error Message: Bad Image when launching MobaXterm under WDAC enforcement. `C:\Users\ADMINI~1\AppData\Roaming\MobaXterm\slash\bin\lib.dll` is displayed.

Root Cause: MobaXterm® is installed with its executable and DLL dependencies in separate directories. The MobaXterm® executable is digitally signed with a valid certificate, but its required DLL dependencies are unsigned.

MobaXterm® is installed with its executable and DLL dependencies in separate directories.

- **MobaXterm® executable**: `C:\Program Files (x86)\Mobatek\MobaXterm`
- **DLL dependencies**: `C:\Users\Administrator\AppData\Roaming\MobaXterm`

The MobaXterm® executable is digitally signed with a valid certificate, but some of its required DLL dependencies are unsigned.

When a WDAC Publisher rule is created for the signed MobaXterm® executable, it allows the executable based on its publisher information. However, the rule does not automatically allow the unsigned DLL dependencies.

When MobaXterm® attempts to load these DLLs, WDAC blocks them because they do not meet the policy's code integrity requirements. As a result, MobaXterm® fails to launch and displays a Bad Image error.

Solution:

1. **Step 1**: Verify the MobaXterm® executable and DLL locations. Verify that the MobaXterm executable and its required DLL dependencies are present in the following directories:
   - **MobaXterm executable**: `C:\Program Files (x86)\Mobatek\MobaXterm`
   - **DLL dependencies**: `C:\Users\Administrator\AppData\Roaming\MobaXterm`

   Verify that the executable is digitally signed and that the required DLL dependencies are unsigned.
2. **Step 2**: Create a publisher rule for MobaXterm™. Use **Task ID: 354** to create a base policy including an allow rule for MobaXterm™.
3. **Step 3**: Use **Task ID: 355** to sign, compile, and publish it.
4. **Step 4**: Use **Task ID: 406** to deploy and enforce the policy.
5. **Result**: The MobaXterm™ installs and launches cleanly.

Product Context and Architecture

BigFix Application Control: Product Overview

System Architecture

BigFix Application Control v2.x: Policy Behavior and Deployment Considerations *(on page 32)*

BigFix Application Control: WDAC Known Limitations *(on page 18)*

Policy Authoring and Management

Manage WDAC Policies on the WDAC Management Node *(on page 49)*

Compile, Sign, and Publish WDAC Policies from the WDAC Management Node *(on page 60)*

Deploy and Refresh WDAC Policies on Target Endpoints *(on page 66)*

Endpoint Verification

Set Up and Initialize WDAC on Target Endpoints *(on page 64)*

Reference and FAQs

BigFix Application Control v2.x: Frequently Asked Questions *(on page 141)*

Notices

This information was developed for products and services offered in the US.

HCL may not offer the products, services, or features discussed in this document in other countries. Consult your local HCL representative for information on the products and services currently available in your area. Any reference to an HCL product, program, or service is not intended to state or imply that only that HCL product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any HCL intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-HCL product, program, or service.

HCL may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

HCL
330 Potrero Ave.
Sunnyvale, CA 94085
USA
Attention: Office of the General Counsel

For license inquiries regarding double-byte character set (DBCS) information, contact the HCL Intellectual Property Department in your country or send inquiries, in writing, to:

HCL
330 Potrero Ave.
Sunnyvale, CA 94085
USA
Attention: Office of the General Counsel

HCL TECHNOLOGIES LTD. PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. HCL may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-HCL websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this HCL product and use of those websites is at your own risk.

HCL may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

HCL
330 Potrero Ave.
Sunnyvale, CA 94085
USA
Attention: Office of the General Counsel

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by HCL under terms of the HCL Customer Agreement, HCL International Program License Agreement or any equivalent agreement between us.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

Information concerning non-HCL products was obtained from the suppliers of those products, their published announcements or other publicly available sources. HCL has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-HCL products. Questions on the capabilities of non-HCL products should be addressed to the suppliers of those products.

Statements regarding HCL's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to HCL, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. HCL, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS," without warranty of any kind. HCL shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:
© (your company name) (year).
Portions of this code are derived from HCL Ltd. Sample Programs.

Trademarks

HCL Technologies Ltd. and HCL Technologies Ltd. logo, and hcl.com are trademarks or registered trademarks of HCL Technologies Ltd., registered in many jurisdictions worldwide.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other product and service names might be trademarks of HCL or other companies.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the HCL website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of HCL.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of HCL.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

HCL reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by HCL, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

HCL MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.